hacker bits

April 2016

new bits

Ah, spring is finally here (in Redmond)…but who's got time to smell the flowers, right? In this jam-packed issue of *Hacker Bits*, we've brought in venture capitalist Patrick McGinnis and asked him some hard questions about becoming an entrepreneur on the side while working full-time.

There's no question that times are tough in Silicon Valley these days, but Heidi Roizen has some solid advice for how startups can stay alive when venture capital dries up. As if life isn't hard enough for startups, Jeff Atwood shows us how the mentality to hire only the best candidates can be hindering their prospects of getting ahead of the pack.

But no worries, life is not all gloom and doom. Don't believe us, just check out Alexander Hogue's hilarious account of how he figured out when his Facebook friends were asleep and awake.

And when you are done perusing this issue, don't forget to check out the awesomeness of our revamped site at hackerbits.com! As always, drop us a line at our site if you like/dislike anything we are doing…or if you just want to chat!

See y'all next month!

— Maureen and Ray

us@hackerbits.com

content bits

April 2016

contributor bits

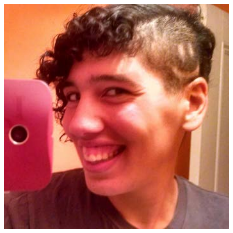
Jeff Atwood
Jeff is a software developer, author, blogger and entrepreneur. He is known for the programming blog Coding Horror, and is the co-founder of the Q&A website Stack Overflow and the Stack Exchange Network.

Ammon Bartram
Ammon was lead video developer at Justin.tv, before co-founding Socialcam in 2011 with Guillaume Luccisano and Michael Seibel. Socialcam was acquired by Autodesk in 2012 for $60m. Now he develops the interview process at Triplebyte, and writes about it.

Jacques Mattheij
Born in 1965, just in time for a front-row-seat for the PC revolution, playing with technology since I was old enough to hold a screwdriver, mechanical stuff, electrical, electronics and finally software. Inventor of live-streaming-video on the web, currently mostly active as interim CEO/CTO and doing technical due-diligence for top-flight VCs.

Heidi Roizen
Heidi is the Operating Partner at DFJ, a lecturer on Entrepreneurship at Stanford, and a recovering entrepreneur.

Trevor McKendrick
Trevor has been featured throughout the web for his work in the Latino market, including the Startup podcast, The Huffington Post, and Fox News. He's now helping other entrepreneurs learn to start and grow their businesses at trevormckendrick.com.



Julia Evans
Julia is a machine learning engineer at Stripe. She likes playing with data, learning systems programming, and making Serious Systems do silly things. You can read her blog at jvns.ca.

Alexander Hogue
Alexander is some kid with a computer. Currently he's a Security Something at Atlassian, which is a little bit like being an adult but with more ice cream. He makes dumb novelty websites (see oneu.se) as a substitute for getting out more.

Matt Mazur
Matt is a developer at Automattic where he focuses on growth for WordPress.com. He currently lives in Orlando, Florida, with his wife and two kids.

Non Umemoto

Non is an indie iOS developer from Japan. He started coding at 2010, and now makes a living by selling Taxnote and simple apps on the App Store. He also writes about Product Design on plumshell.com.

Daniel Stenberg

Daniel is an Internet protocol geek employed by Mozilla and the founder and lead developer in the curl project.

Kenneth Reitz

Kenneth is a software engineer, photographer, and electronic musician. He has a deep love for Python and is well known for creating Requests: HTTP for Humans, and many other open source projects.

Ray Li

Curator

Ray is a software engineer and data enthusiast who has been blogging at rayli.net for over a decade. He loves to learn, teach and grow. You'll usually find him wrangling data, programming and lifehacking.

Maureen Ker

Editor

Maureen is an editor, writer, enthusiastic cook and prolific collector of useless kitchen gadgets. She is the author of 3 books and 100+ articles. Her work has appeared in the *New York Daily News*, and various adult and children's publications.

Opinion

We hire the best, just like everyone else

By JEFF ATWOOD

One of the most common pieces of advice you'll get as a startup is this:

> **Only hire the best.** *The quality of the people that work at your company will be one of the biggest factors in your success – or failure.*

I've heard this advice over and over and over at startup events, to the point that I got a little sick of hearing it. It's not wrong. Putting aside the fact that every single other startup in the world who heard this same advice before you is already out there frantically doing everything they can to hire all the best people out from under you and everyone else, it is superficially true. A company staffed by a bunch of people who don't care about their work and aren't good at their jobs isn't exactly poised for success. But in a room full of people giving advice to startups, nobody wants to talk about the elephant in that room:

> *It doesn't matter how good the people are at your company when you happen to be working on the wrong problem, at the wrong time, using the wrong approach.*

Most startups, statistically speaking, are going to fail.

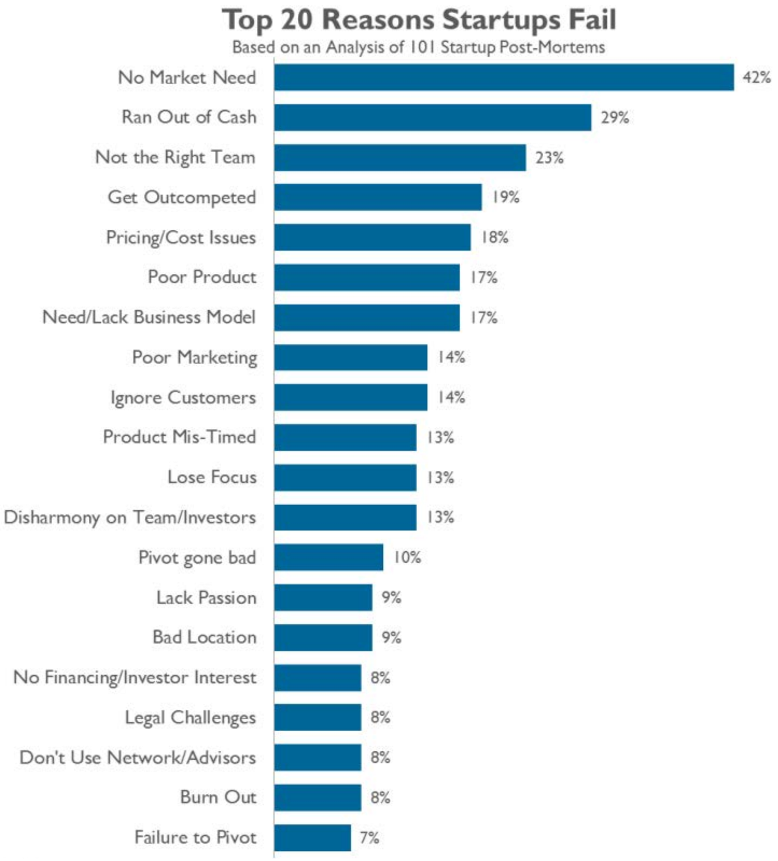
And they will fail regardless of whether they hired "the best" due to circumstances largely beyond their control. So in that context does maximizing for the best possible hires really make sense?

Given the risks, I think maybe "hire the nuttiest risk junkie adrenaline addicted has-ideas-so-crazy-they-will-never-work people you can find" might actually be more practical startup advice. (Actually, now that I think about it, if that describes you, and you have serious Linux, Ruby, and JavaScript chops, perhaps you should email me.)

Okay, the goal is to increase your *chance* of success, ~~however small it may be~~, therefore you should strive to hire the best. Seems reasonable, even noble in its way. But this pursuit of the best unfortunately comes with a serious dark side. Can anyone even tell me what "best" is? By what metrics? Judged by which results? How do we measure this? Who among us is suitable to judge others as the best at… what, exactly? Best is an extreme. Not pretty good, not very good, not excellent, but aiming for the *crème de la crème*, the top 1% in the industry.

> *The real trouble with using a lot of mediocre programmers instead of a couple of good ones is that no matter how long they work, they never produce something as good as what the great programmers can produce.*

Pursuit of this extreme means hiring anyone less than the best becomes unacceptable, even harmful:



Credit: www.cbinsights.com

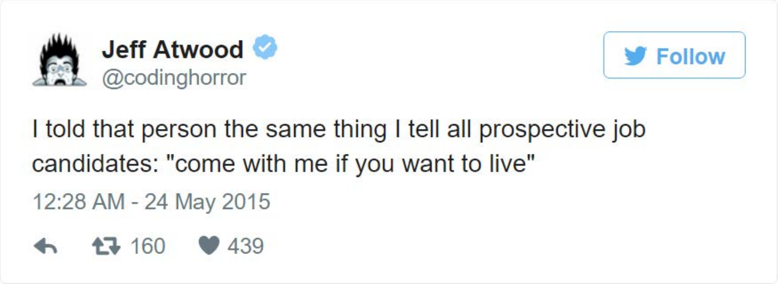
In the Macintosh Division, we had a saying, “A players hire A players; B players hire C players” – meaning that great people hire great people. On the other hand, mediocre people hire candidates who are not as good as they are, so they can feel superior to them. (If you start down this slippery slope, you’ll soon end up with Z players; this is called The Bozo Explosion. It is followed by The Layoff.) — Guy Kawasaki

There is an opportunity cost to keeping someone when you could do better. At a startup, that opportunity cost may be the difference between success and failure. Do you give less than full effort to make your enterprise a success? As an entrepreneur, you sweat blood to succeed. Shouldn’t you have a team that performs like you do? Every person you hire who is not a top player is like having a leak in the hull. Eventually you will sink. — Jon Soberg

Why am I so hardnosed about this? It’s because it is much, much better to reject a good candidate than to accept a bad candidate. A bad candidate will cost a lot of money and effort and waste other people’s time fixing all their bugs. Firing someone you hired by mistake can take months and be nightmarishly difficult, especially if they decide to be litigious about it. In some situations it may be completely impossible to fire anyone. Bad employees demoralize the good employees. And they might be bad programmers but really nice people or maybe they really need this job, so you can’t bear to fire them, or you can’t fire them without pissing everybody off, or whatever. It’s just a bad scene.

On the other hand, if you reject a good candidate, I mean, I guess in some existential sense an injustice has been done, but, hey, if they’re so smart, don’t worry, they’ll get lots of good job offers. Don’t be afraid that you’re going to reject too many people and you won’t be able to find anyone to hire. During the interview, it’s not your problem. Of course, it’s important to seek out good candidates. But once you’re actually interviewing someone, pretend that you’ve got 900 more people lined up outside the door. Don’t lower your standards no matter how hard it seems to find those great candidates.

— Joel Spolsky



Jeff Atwood
@codinghorror

I told that person the same thing I tell all prospective job candidates: "come with me if you want to live"

12:28 AM - 24 May 2015

160 439

It is better to reject a good applicant every single time than accidentally accept one single mediocre applicant.

I don't mean to be critical of anyone I've quoted. I love Joel, we founded Stack Overflow together, and his advice about interviewing and hiring remains some of the best in the industry. It's hardly unique to express these sort of opinions in the software and startup field. I could have cited two dozen different articles and treatises about hiring that say the exact same thing: aim high and set out to hire the best, or don't bother.

This risk of hiring not-the-best is so severe, so existential a crisis to the very survival of your company or startup that the hiring process has to become highly selective, even arduous. **It is better to reject a good applicant *every single time* than accidentally accept one single mediocre applicant.** If the interview process produces literally anything other than unequivocal "Oh my God, this person is unbelievably talented, we have to hire them", from every single person they interviewed with, right down the line, then it's an automatic NO HIRE. Every time.

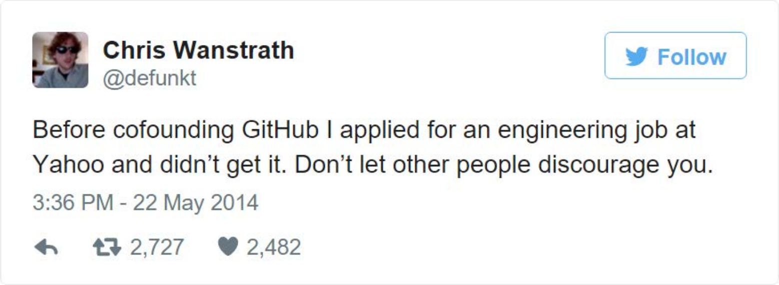
This level of strictness always made me uncomfortable. I'm not going to lie, it starts with my own selfishness. I'm pretty sure I wouldn't get hired at big, famous companies with legendarily difficult technical interview processes because, you know, they only hire the best. I don't *think I am one of the best*. More like cranky, tenacious, and outspoken, to the point that I wake up most days not even wanting to work with myself.

If your hiring attitude is that it's better to be possibly wrong a hundred times so you can be absolutely right one time, you're going to be primed to throw away a lot of candidates on pretty thin evidence.

Perhaps worst of all, if the interview process is predicated on zero doubt and total confidence … maybe this candidate doesn't feel right because they don't look like you, dress like you, think like you, speak like you, or come from a similar background as you? Are you **accidentally maximizing for hidden bias?**

One of the best programmers I ever worked with was Susan Warren, an ex-Microsoft engineer who taught me about the People Like Us problem, way back in 2004:

> *I think there is a real issue around diversity in technology (and most other places in life). I tend to think of it as the PLU problem. Folks (including MVPs) tend to connect best with folks most like them ("People Like Us"). In this case, male MVPs pick other men to become MVPs. It's just human nature.*
>
> *As one reply notes, diversity is good. I'd go as far as to say it's awesome, amazing and priceless. But it's hard to get to the classic chicken and egg problem if you rely on your natural tendencies alone. In that case, if you want more female MVPs to be invited, you need more female MVPs. If you want more Asian-American MVPs to be invited, you need more Asian-American MVPs, etc. And the (cheap) way to break a new group in is via quotas.*
>
> *IMO, building diversity via quotas is bad because they are unfair. Educating folks on why diversity is awesome and how to build it is the right way to go, but also far more costly.*

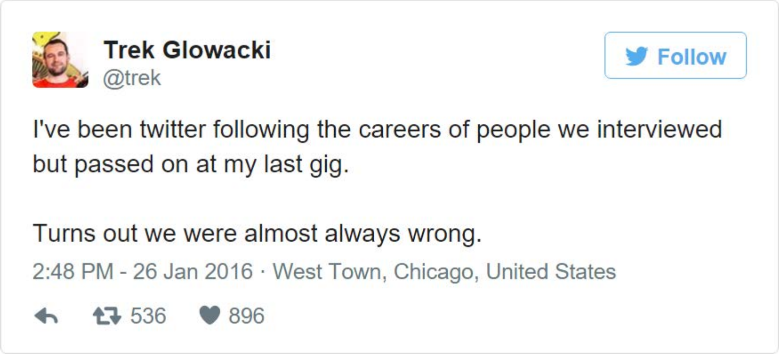


Chris Wanstrath
@defunkt

Before cofounding GitHub I applied for an engineering job at Yahoo and didn't get it. Don't let other people discourage you.

3:36 PM - 22 May 2014

2,727 2,482



Trek Glowacki
@trek

I've been twitter following the careers of people we interviewed but passed on at my last gig.

Turns out we were almost always wrong.

2:48 PM - 26 Jan 2016 · West Town, Chicago, United States

536 896

Susan was (and is) amazing. I learned so much working under her, and a big part of what made her awesome was that she was very much Not Like Me. But how could I have appreciated that before meeting her? The fact is that as human beings, we tend to prefer what's comfortable, and what's most comfortable of all is … well, People Like Us. The effect can be shocking because it's so subtle, so unconscious – and yet, surprisingly strong:

- Baseball cards held by a black hand consistently sold for twenty percent less than those held by a white hand.
- Using screens to hide the identity of auditioning musicians increased women's probability of advancing from preliminary orchestra auditions by fifty percent.
- Denver police officers and community members were shown rapidly displayed photos of black and white men, some holding guns, some holding harmless objects like wallets, and asked to press either the "Shoot" or "Don't Shoot" button as fast as they could for each image. Both the police and community members were three times more likely to shoot black men.

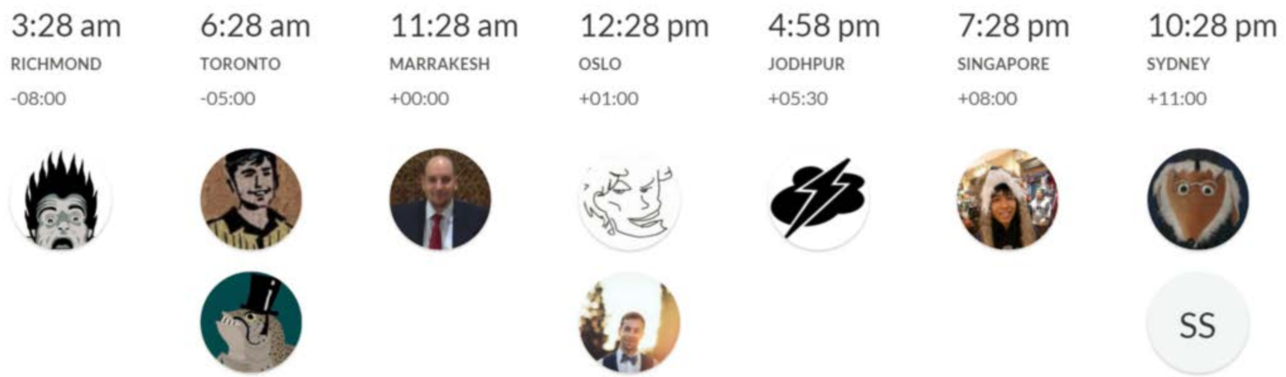
It's not intentional, it's never intentional. That's the problem. I think our industry needs to shed this old idea that it's OK, even *encouraged* to turn away technical candidates for anything less than absolute 100% confidence at every step of the interview process. Because when you do, **you are accidentally optimizing for implicit bias**. Even as a white guy who probably fulfills every stereotype you can think of about programmers, and who is in fact wearing an "I Rock at Basic" t-shirt while writing this very blog post*, that's what has always bothered me about it, more than the strictness. If you care at all about diversity in programming and tech, on any level, this hiring approach is not doing anyone any favors, and hasn't been. For years.

I know what you're thinking.

> *Fine, Jeff, if you're so smart, and "hiring the best" isn't the right strategy for startups, and maybe even harmful to our field as a whole, what should be doing?*

Well, I don't know, exactly. I may be the wrong person to ask because I'm also a big believer in *geographic* diversity on top of everything else. Here's what the composition of the current Discourse team looks like. (figure below)

I would argue, quite strongly and at some length, that if you want better diversity in the field, perhaps a good starting point is **not demanding that all your employees live within a tiny 30 mile radius of San Francisco or Palo Alto.** There's a whole wide world of Internet out there, full of amazing programmers at every level of talent and ability. Maybe broaden your horizons a little, even stretch said horizons outside the United States, if you



3:28 am	6:28 am	11:28 am	12:28 pm	4:58 pm	7:28 pm	10:28 pm
RICHMOND	TORONTO	MARRAKESH	OSLO	JODHPUR	SINGAPORE	SYDNEY
-08:00	-05:00	+00:00	+01:00	+05:30	+08:00	+11:00

Look beyond People Like Us and imagine what the world of programming could look like in 10, 20 or even 50 years...

can imagine such a thing.

I know hiring people is difficult, even with the very best of intentions and under ideal conditions, so I don't mean to trivialize the challenge. I've recommended plenty of things in the past, a smorgasboard of approaches to try or leave on the table as you see fit:

- On Interviewing Programmers
- The Non-Programming Programmer
- How to Hire a Programmer
- The years of experience myth

… but the one thing I keep coming back to, that I believe has enduring value in almost all situations, is the audition project:

> *The most significant shift we've made is requiring every final candidate to work with us for three to eight weeks on a contract basis. Candidates do real tasks alongside the people they would actually be working with if they had the job. They can work at night or on weekends, so they don't have to leave their current jobs; most spend 10 to 20 hours a week working with Automattic, although that's flexible. (Some people take a week's vacation in order to focus on the tryout, which is another viable option.) The goal is not to have them finish a product or do a set amount of work; it's to allow us to quickly and efficiently assess whether this would be a mutually beneficial relationship. They can size up Automattic while we evaluate them.*

What I like about audition projects:

- It's real, practical work.
- They get paid. (Ask yourself who gets "paid" for a series of intensive interviews that lasts multiple days? Certainly not the candidate.)
- It's healthy to structure your work so that small projects like this can be taken on by outsiders. If you can't onboard a potential hire, you probably can't onboard a new hire very well either.
- Interviews, no matter how much effort you put into them, are so hit and miss that the only way to figure out if someone is really going to work in a given position is to **actually work with them.**

Every company says they want to hire the best. Anyone who tells you they know how to do that is either lying to you or to themselves. But I can tell you this: the companies that really do hire the best people in the world certainly don't accomplish that by hiring from the same tired playbook every other company in Silicon Valley uses.

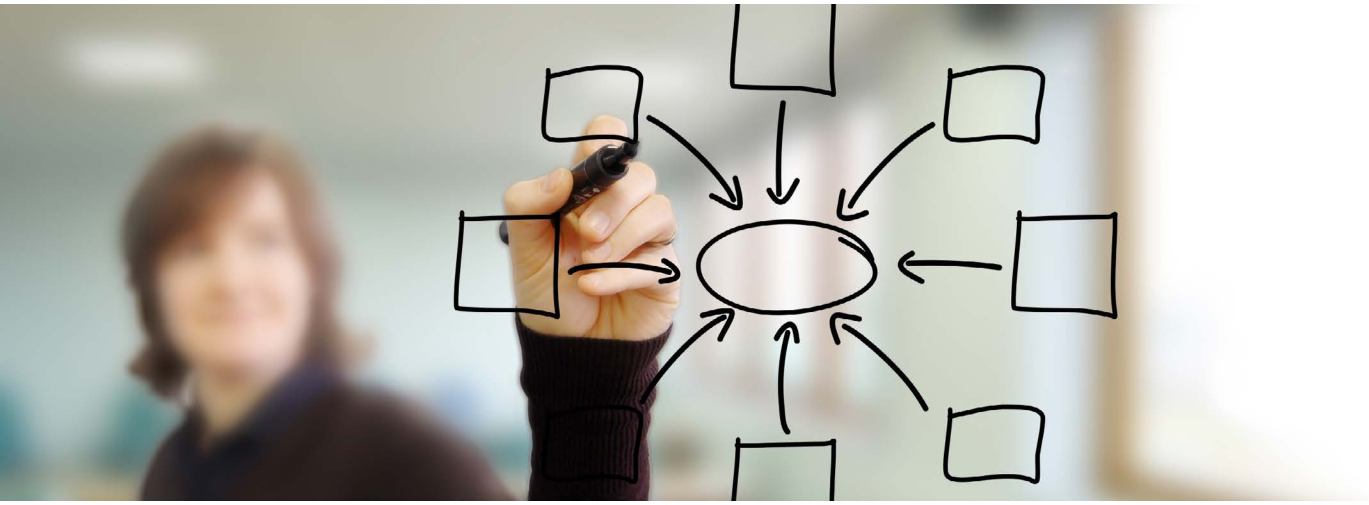
Try different approaches. Expand your horizons. Look beyond People Like Us and imagine what the world of programming could look like in 10, 20 or even 50 years – and help us move there by hiring to make it so. ■

* And for the record, I really do rock at BASIC.

Reprinted with permission of the original author. First appeared at blog.codinghorror.com.

How to pass a programming interview

By AMMON BARTRAM



Being a good programmer has a surprisingly small role in passing programming interviews. To be a productive programmer, you need to be able to solve large, sprawling problems over weeks and months. Each question in an interview, in contrast, lasts less than one hour.

To do well in an interview, then, you need to be able to solve small problems quickly, under duress, while explaining your thoughts clearly. This is a different skill. On top of this, interviewers are often poorly trained and inattentive (they would rather be programming), and ask questions far removed from actual work. They bring bias, pattern matching, and a lack of standardization.

Running Triplebyte, I see this clearly. We interview engineers without looking at resumes, and fast-track them to on-sites at YC companies. We've interviewed over 1,000 programmers in the last nine months. We focus heavily on practical programming, and let candidates pick one of several ways to be evaluated. This means we work with many (very talented) programmers without formal CS training. Many of these people do poorly on interviews. They eat large sprawling problems for breakfast, but they balk at 45-min algorithm challenges.

The good news is that interviewing is a skill that can be learned. We've had success teaching candidates to do better on interviews. Indeed, the quality that most correlates with a Triplebyte candidate passing interviews at YC companies is not raw talent, but rather diligence.

I fundamentally do not believe that good programmers should have to learn special interviewing skills to do well on interviews. But the status quo is what it is. We're working at Triplebyte to change this. If you're interested in what we're doing, we'd love to have you to check out our process. In the meantime, if you do want to get better at interviewing, this article describes how we think you can most effectively do so.

About 50% of the Triplebyte candidates who fail interviews at companies fail for non-technical reasons.

1. Be enthusiastic

Enthusiasm has a huge impact on interview results. About 50% of the Triplebyte candidates who fail interviews at companies fail for non-technical reasons. This is usually described by the company as a “poor culture fit”. Nine times out of ten, however, culture fit just means enthusiasm for what a company does. Companies want candidates who are excited about their mission. This carries as much weight at many companies as technical skill. This makes sense. Excited employees will be happier and work harder.

The problem is that this can be faked. Some candidates manage to convince every company they talk to that it’s their dream job, while others (who are genuinely excited) fail to convince anyone. We’ve seen this again and again. The solution is for everyone to get better at showing their enthusiasm. This is not permission to lie. But interviewing is like dating. No one wants to be told on a first date that they are one option among many, even though this is usually the case. Similarly, most programmers just want a good job with a good paycheck. But stating this in an interview is a mistake. The best approach is to prepare notes before an interview about what you find exciting about the company, and bring this up with each interviewer when they ask if you have any questions. A good source of ideas is to read the company’s recent blog posts and press releases, and note the ones you find exciting.

This idea seems facile. I imagine you are nodding along as you read this. But (as anyone who has ever interviewed can tell you) a surprisingly small percentage of applicants do this. Carefully preparing notes on why you find a company exciting really will increase your pass rate. You can even reference the notes during the interview. Bringing prepared notes shows preparation.

2. Study common interview concepts

A large percentage of interview questions feature data structures and algorithms. For better or worse, this is the truth. We gather question details from our candidates who interview at YC companies (we’ll be doing an in-depth analysis of this data in a future article), and algorithm questions make up over 70% of the questions that are asked. You do not need to be an expert, but knowing the following list of algorithms and data structures will help at most companies.

- Hash tables
- Linked lists
- Breadth-first search, depth-first search
- Quicksort, merge sort
- Binary search
- 2D arrays
- Dynamic arrays
- Binary search trees
- Dynamic programming
- Big-O analysis

Depending on your background, this list may look trivial, or may look totally intimidating. That’s exactly the point. These are concepts that are far more common in interviews than they are in production web programming. If you’re self-taught or years out of school and these concepts are not familiar to you, you will do better in interviews if you study them. Even if you do know these things, refreshing your knowledge will help.

A startlingly high percentage of interview questions reduce to breadth-first search or the use of a hash table to count uniques. You need to be able to write a BFS cold, and you need to understand how a hash table is implemented.

Learning these things is not

as hard as many of the people we talk to fear. Algorithms are usually described in academic language, and this can be off-putting. But at its core, nothing on this list is more complicated than the architecture of a modern web app. If you can build a web app (well), you can learn these things.

The resource that I recommend is the book *The Algorithm Design Manual* by Steven Skiena. Chapters 3 through 5 do a great job of going over this material, in a straightforward way. It does use C and some math syntax, but it explains the material well. Coursera also has several good algorithms courses. This one, in particular, focuses on the concepts that are important in interviews.

Studying algorithms and data structures helps not only because the material comes up in interviews, but also because the approach to problems taken in an algorithm course is the same approach that works best in interviews. Studying algorithms will get you in an interview mindset.

3. Get help from your interviewer

Interviewers help candidates. They give hints, they respond to ideas, and they generally guide the process. But they don't help all candidates equally. *Some programmers are able to extract significant help, without the interviewer holding it against them.* Others are judged harshly for any hints they are given. You want to be helped.

This comes down to process and communication. If the interviewer likes your process and you communicate well with them, they will not mind helping. You can make this more likely by following a careful process. The steps I recommend are:

1. Ask questions
2. Talk through a brute-force solution
3. Talk through an optimized solution
4. Write code

After you are asked an interview question, start by clarifying what was asked. This is the time to be pedantic. Clarify every ambiguity you can think of. Ask about edge cases. Bring up specific examples of input, and make sure you are correct about the expected output. Ask questions even if you're almost sure you know the answers. This is useful because it gives you a chance to come up with edge cases and fully spec the problem (seeing how you handle edge-cases is one of the main things that interviewers look for when evaluating an interview), and also because it gives you a minute to collect your thoughts before you need to start solving the problem.

Next, you should talk through the simplest brute-force solution to the problem that you can think of. You should talk, rather than jump right into coding, because you can move faster when talking, and it's more engaging for the interviewer. *If the interviewer is engaged, they will step in and offer pointers.* If you retreat into writing code, however, you'll miss this opportunity.

Candidates often skip the brute-force step, assuming that the brute-force solution to the problem is too obvious, or wrong. This is a mistake. Make sure that you always give a solution to the problem you've been asked (even if it takes exponential time, or an NSA super computer). When you've described a brute-force solution, ask the interviewer if they would like you to implement it, or come up with more efficient solution. Normally they will tell you to come up with a more efficient solution.

The process for the more efficient solution is the same as for the brute force. Again talk, don't write code, and bounce ideas off of the interviewer. Hopefully, the question will be

Some programmers are able to extract significant help, without the interviewer holding it against them.

similar to something you've seen, and you'll know the answer. If that is not the case, it's useful to think of what problems you've seen that are most similar, and bring these up with the interviewer. Most interview questions are slightly-obscured applications of classic CS algorithms. The interviewer will often guide you to this algorithm, but only if you begin the process.

Finally, after both you and your interviewer agree that you have a good solution, you should write your code. Depending on the company, this may be on a computer or a whiteboard. But because you've already come up with the solution, this should be fairly straightforward. For extra points, ask your interviewer if they would like you to write tests.

...you may also encounter system design questions. Companies seem to like these especially for more experienced candidates.

4. Talk about trade-offs

Programming interviews are primarily made up of programming questions, and that is what I have talked about so far. However, you may also encounter system design questions. Companies seem to like these especially for more experienced candidates. In a system design question, the candidate is asked how he or she would design a complex real-world system. Examples include designing Google maps, designing a social network, or designing an API for a bank.

The first observation is that answering system design questions requires some specific knowledge. Obviously no one actually expects you to design Google maps (that took a lot of people a long time). But they do expect you to have some insight into aspects of such a design. The good news is that these questions usually focus on web backends, so you can make a lot of progress by reading about this area. An incomplete list of things to understand is:

- HTTP (at the protocol level)
- Databases (indexes, query planning)
- CDNs
- Caching (LRU cache, memcached, redis)
- Load balancers
- Distributed worker systems

You need to understand these concepts. But more importantly, you need to understand how they fit together to form real systems. The best way to learn this is to read about how other engineers have used the concepts. The blog High Scalability is a great resource for this. It publishes detailed write-ups of the back-end architecture at real companies. You can read about how every concept on the list above is used in real systems.

Once you've done this reading, answering system design questions is a matter of process. Start at the highest level, and move downward. At each level, ask your interviewer for specifications (should you suggest a simple starting point, or talk about what a mature system might look like?) and talk about several options (applying the ideas from your reading). Discussing tradeoffs in your design is key. Your interviewer cares less about whether your design is good in itself, and more about whether you are able to talk about the trade-offs (positives and negatives) of your decisions. Practice this.

5. Highlight results

The third type of question you may encounter is the experience question. This is where the interviewer asks you to talk about a programming project that

Programmers too focused on interesting tech is an anti-pattern that companies screen against.

you completed in the past. The mistake that many engineers make on this question is to talk about a technically interesting side-project. Many programmers choose to talk about implementing a neural network classifier, or writing a Twitter grammar bot. These are bad choices because it's very hard for the interviewer to judge their scope. Many candidates exaggerate simple side projects (sometimes that never actually worked), and the interviewer has no way to tell if you are doing this.

The solution is to choose a project that produced results, and highlight the results. This often involves picking a less technically interesting project, but it's worth it. Think (ahead of time) of the programming you've done that had the largest real-world impact. If you've written a iOS game, and 50k people have downloaded it, the download number makes it a good option. If you've written an admin interface during an internship that was deployed to the entire admin staff, the deployment makes it a good thing to talk about. Selecting a practical project will also communicate to the company that you focus on actual work. Programmers too focused on interesting tech is an anti-pattern that companies screen against (these programmers are sometimes not productive).

6. Use a dynamic language, but mention C

I recommend that you use a dynamic language like Python, Ruby or JavaScript during interviews. Of course, you should use whatever language you know best. But we find that many people try interviewing in C , C++ or Java, under the impression these are the "real' programming languages. Several classic books on interviewing recommend that programmers choose Java or C++. At startups at least, we've found that this is bad advice. Candidates do better when using dynamic languages. This is true, I think, because of dynamic languages' compact syntax, flexible typing, and list and hash literals. They are permissive languages. This can be a liability when writing complex systems (a highly debatable point), but it's great when trying to cram binary search onto a whiteboard.

No matter what language you use, it's helpful to mention work in other languages. An anti-pattern that companies screen against is people who only know one language. If you do only know one language, you have to rely on your strength in that language. But if you've done work or side-projects in multiple languages, be sure to bring this up when talking to your interviewers. If you have worked in lower-level languages like C, C++, Go, or Rust, talking about this will particularly help.

Java, C# and PHP are a problematic case. As we described in our last blog post, we've uncovered bias against these languages in startups. We have data showing that programmers using these languages in the interview pass at a lower rate. This is not fair, but it is the truth. If you have other options, I recommend against using these languages in interviews with startups.

7. Practice, practice, practice

You can get much better at interviewing by practicing answering questions. This is true because interviews are stressful, but stress harms performance. The solution is practice. Interviewing becomes less stressful with exposure. This happens naturally with experience. Even within a single job search, we find that candidates often fail their initial interviews, and then pass more as their confidence builds. If stress is something you struggle with, I recommend that you jumpstart this process by *practicing interview stress*. Get a list of interview questions (the book *Cracking the Coding Interview* is one good source) and solve them. Set a 20-minute timer on each question, and race to answer. Practice writing the answers on a whiteboard (not all companies require this, but it's the worst case, so you should practice it). A pen on paper is a pretty good simulation of a whiteboard. If you have friends who can help you prepare, taking turns interviewing each other is great. Reading a lot of interview questions has the added benefit of providing you ideas to use when in actual interviews. A surprising number of questions are re-used (in full or in part).

Even experienced (and stress-free) candidates will benefit from this. Interviewing is a fundamentally different skill from working as a programmer, and it can atrophy. But experienced programers often (reasonably) feel that they should not have to prepare for interviews. They study less. This is why junior candidates often actually do better on interview questions than experienced candidates. Companies know this, and, paradoxically, some tell us they set lower bars on the programming questions for experienced candidates.

8. Mention credentials

Credentials bias interviewers. Triplebyte candidates who have worked at a top company or studied at a top school go on to pass interviews at a 30% higher rate than programmers who don't have these credentials (for a given level of performance on our credential-blind screen). I don't like this. It's not meritocratic and it sucks, but if you have these credentials, it's in your interest to make sure that your interviewers know this. You can't trust that they'll read your resume.

9. Line up offers

If you've ever read fund-raising advice for founders, you'll know that getting the 1st VC to make an investment offer is the hardest part. Once you have one offer, more come pouring in. The same is true of job offers. If you already have an offer, be sure to mention this in interviews. Mentioning other offers in an interview heavily biases the interviewer in your favor.

This brings up the strategy of making a list of the companies you're interested in, and setting up interviews in *reverse order of interest*. Doing well earlier in the process will increase your probability of getting an offer from your number one choice. You should do this.

Conclusion

Passing interviews is a skill. Being a great programmer helps, but it's only part of the picture. Everyone fails some of their interviews, and preparing properly can help everyone pass more. Enthusiasm is paramount, and research helps with this. As many programmers fail for lacking enthusiasm as those who fail for technical reasons. Interviewers help candidates during interviews, and if you follow a good process and communicate clearly, they will help you. Practice always helps. Reading lots of interview questions and inuring yourself to interview stress will lead to more offers.

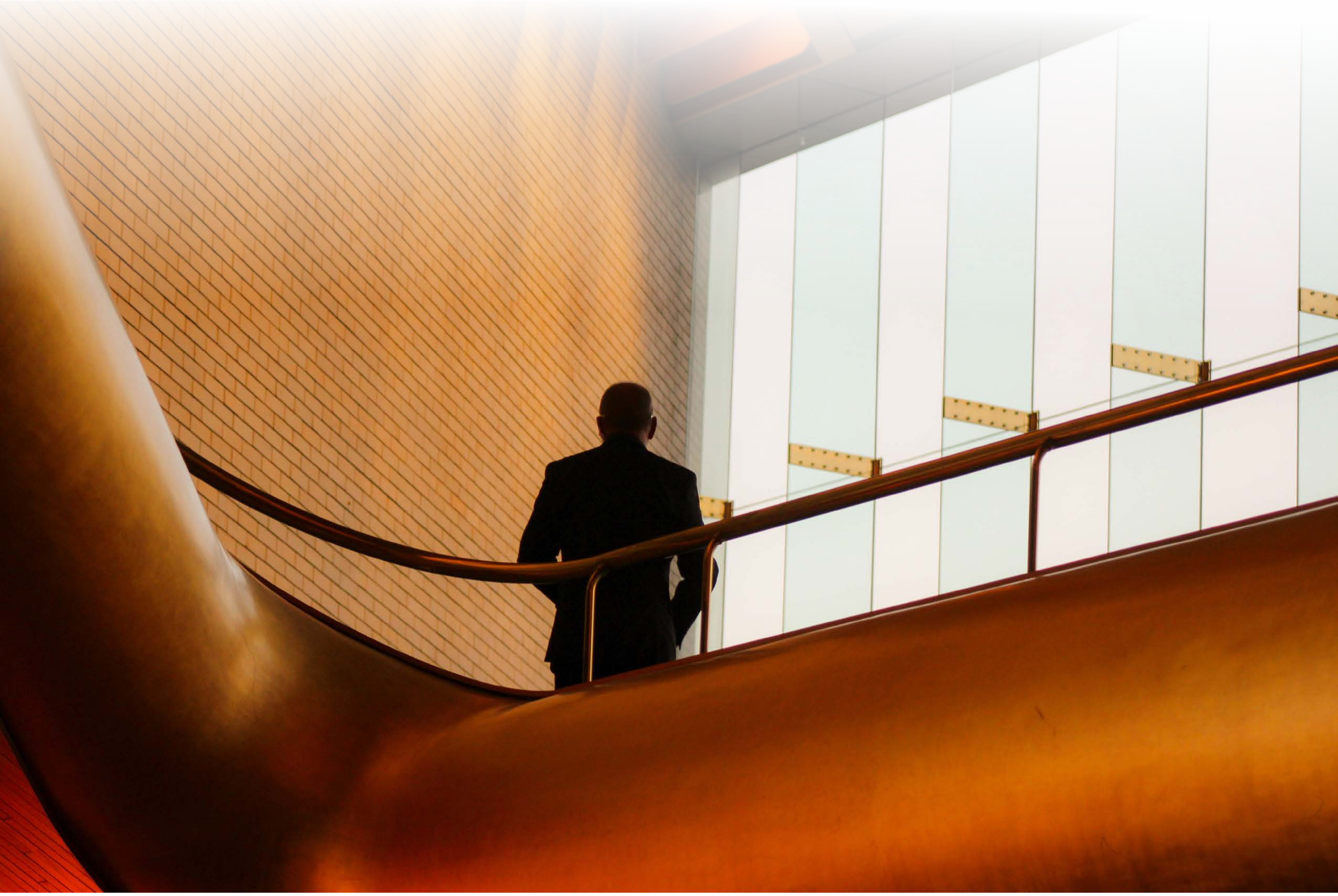
This situation is not ideal. Preparing for interviews is work, and forcing programmers to learn skills other than building great software wastes everyone's time. Companies should improve their interview processes to be less biased by academic CS, memorized facts, and rehearsed interview processes. This is what we're doing at Triplebyte. We help programmers get jobs without looking at resumes. We let programmers pick one of several areas in which to be evaluated, and we study and improve our process over time. We'd love to help you get a job at a startup, without jumping through these hoops. You can get started here. But the status quo is what it is. Until this changes, programmers should know how to prepare. ■

Reprinted with permission of the original author. First appeared at blog.triplebyte.com.

Opinion

CEO

By JACQUES MATTHEIJ



As a CEO, you are ultimately responsible for whatever happens to and in your company...

In this article I'm going to try to give potential and actual CEOs (and other C level execs to some extent) some actionable advice when it comes to how they conduct themselves, in the hope that this will at some future date save some of you a great deal of misery and in some cases, charges of misconduct or worse.

My hope is that this will somehow help save a few companies by causing you, the CEO, to change direction before it is too late to do so.

As seen from the outside, the job of a CEO seems to be composed of equal components motivator, ambassador, networker and the person that 'sets the example.' And that's all true but there is also another side to being a CEO, one that is much more formalized and governed by all kinds of rules.

In my practice I come across all kinds of companies (and all sizes) and unfortunately sometimes I also come across companies that are sick, some so sick that they will not survive. The reason I'm writing this article is that recently I again was sent in to try to save a company from going under and I feel that if the CEO of that company had read an article like this maybe six months ago, this whole disaster could have been avoided.

Companies going under is a real pity, not only for the founders and investors. The real losers in those cases are the employees, the suppliers and the customers. They bet on the wrong horse for something that will now materially affect their lives. They feel (in some cases rightly so) that their trust has been misplaced, and in some cases, they will feel duped or defrauded.

When the company is doing ok, there is money in the bank. When you're hiring and the sun is shining, anybody can be CEO of a company. But when the bank accounts are running dry, when you have to fire people and when it seems as if there is just no end to the bad news, it really matters who is in the driving seat. It seems such a great thing, to be the master of a company but it is a double edged sword. With all that freedom and executive power comes the flip side of that coin: executive responsibility.

Companies being sick is in and of itself not a rare thing, but what bothers me is that, in many of these cases, the company is much sicker than it has to be. The wounds are, to a large extent, self-inflicted, and, if only someone had had the presence of mind to change direction when it was still possible, the company might have survived.

Before we go any further, let's take a look at what it is that a CEO actually does, what room to maneuver there is, what you're obliged to do, what you are allowed to do and what you're explicitly forbidden to do. Once that is out of the way, we'll get to some more general advice.

The actual legal situation of what it means to be a CEO varies from country to country. I've tried to keep the text presented here general enough that you should get some (or significant) mileage out of it no matter where you are. But if you currently are CEO of a company and you are yourself unclear about your role or if you feel that anything here clashes with the way you see things, then I would strongly suggest you hire a competent lawyer in your jurisdiction for a couple of hours. Let them explain as clearly as possible (and where applicable in your own language) what the local situation is.

Quite a few companies that end up dead do not die of external causes. Plenty of them are murdered from the inside, either by bad decisions made on purpose, by inaction, malice or incompetence and it is frustrating to see this happen when it could have been so easily avoid-

...it's your ass on the line when things go wrong...

ed. For every company that has ever died leaving behind a pile of unpaid bills and ruined lives, there was a point in time when it could have changed direction, when there still were options to affect a gentle landing rather than a crash.

No matter where you are, remember this above everything else: As a CEO, you are ultimately responsible for whatever happens to and in your company, and in some countries that responsibility translates into personal liability if and when things go wrong, especially if it is determined that you did not play by the local variation of the rules and this can have far reaching consequences. In some countries, depending on the severity of the situation, this can become a criminal matter so be careful not to assume a role that you do not fully understand. The implications of CEO mistakes or misconduct can be very serious indeed.

What you are obliged to do

As a CEO you are obliged to keep the company on the right side of the law.

So, for example, if there is some kind of lucrative shortcut to making a lot of money but it is illegal in your jurisdiction to do so, don't do it. Even if it means the difference between going under and surviving the long term fallout of such a decision is potentially much worse than shutting your company down gracefully.

As a CEO, you have to avoid at all costs entering into obligations that you know (or should have known) the company cannot fulfill.

So don't hire new people when you know that you are low on funds and don't order goods or services that you know you will not be able to pay for when the invoice is due.

As a CEO you have to be realistic about your expectations for the near future (say 6 months out) in your day-to-day decision making. Hope for the best but plan for the worst and make sure that the company does not become insolvent (that's a pretty word for being unable to fulfill its financial obligations). Do not assume that promises of future contracts, payments or funding will materialize until they do (that is, the money lands in your bank account) and do not let hope guide your decision making — stick to verifiable facts.

As a CEO you are obliged to decide on any major issues facing the company. Inaction, or postponing a decision for so long that a situation spirals out of control is generally speaking not an option. If that is the way you want to deal with problems then it may be better to step down. The rest of the company will look to your office for guidance and resolution. It is not proper for a CEO to meet challenges with a lack of decisiveness. The usual result of a lack of decision-making power will be that small problems become large problems that will eventually cause the company to go under.

So, for example if the company is being sued, do not ignore the suit; if a big customer leaves, analyze the impact of this and adjust course if necessary while you still can.

As a CEO, you determine the general direction the company will move in, the overall strategy and how that strategy will be implemented. You allocate company resources in order to achieve these goals. In larger companies, it is common that you will seek input from others and if necessary from outside counsel or the board.

Even so, no matter who advises you and how insistent they are, in the end the decisions are yours and yours alone. If there is a conflict between the direction that you think the company should go in or if there is second guessing of your decisions, in the end your word is the one that matters.

Nobody (and this is a very important point) can make you do something that you are not comfortable with. If there is a conflict about important matters, the board (assuming you have one) or depending on the situation, the shareholders can fire you. But, they cannot make you do anything at all that you do not agree with.

This is especially important when a company gets into hot water (for instance, when it is near insolvency or when the direction others want you to go in conflicts with the law or with what you feel is proper behavior). It is extremely important because even though everybody else may have ideas on what should be done, the final responsibility for those decisions lies with you. In plain English: it's your ass on the line when things go wrong, so do not allow yourself to be pushed or intimidated into making decisions that will come back to haunt you later.

So if the board tells you that they want to see you hire more people to push a project forward faster but your bank account does not allow it, secure funding first, then hire more people.

What you are allowed to do

As a CEO, you are allowed to hand in your resignation at any point in time. Even though you are allowed to hand in your resignation, sometimes it is smart not to.

For instance, if the company is in hot water and you still have the ability to make things better for entities (people, suppliers, the taxman) the company owes money to, then you probably should do so. You hand over the company to your successor in the best shape possible with a clear eye towards discharging your responsibilities with propriety. The captain should not leave the bridge of the ship until he has done his very best to save as many lives as possible. Then and only then does he get to save his own skin by jumping overboard.

A weak CEO will throw up their hands in frustration and leave when they are needed most. That in my book is a bad mark against them. But that said, nobody can force you to be CEO of anything. If you are not comfortable with your role and feel that you are either not qualified or no longer able to function (for instance, due to board or shareholder pressure), it is *much* better to step down than to make bad decisions or to continue in a direction that you are personally not comfortable with.

If the board or shareholders want you to go in a direction that you are not comfortable with, it is also perfectly ok to simply tell them you won't do it. They may fire you, but that's much better than ordering the company to set course in a way that does not sit well with you or that you feel is contrary to good governance.

As a CEO you are allowed to bind the company — your signature will have the power to enter the company into obligations. Of course the company will have to be able to fulfill those obligations. Examples of the kinds of obligations you can enter the company into include contracts with customers, ordering goods and services from suppliers, entering into employment contracts with employees and so on.

So (taking into account the ability of the company to perform), you are free to enter into contracts with others. Keep in mind though that some articles of incorporation put limits on the freedom of the CEO to enter into contracts, there may also be upper limits when it comes to amounts or certain classes of decisions. For instance, the articles of incorporation may say that the CEO cannot enter into contractual obligations exceeding 50K, or there may be provisions related to relocating the company, putting liens on company property (intellectual or otherwise), starting legal action and so on without specific approval of either the board and/or the shareholders.

A fine point of order here is that even if you exceed your authority, the company is *still* bound by your signature, and you may then have a real problem with the board or the shareholders (and may even be liable).

As a CEO you determine the general direction the company is headed in and in the 'org chart,' your name comes at the top. All

...do not allow yourself to be pushed or intimidated into making decisions that will come back to haunt you later.

Finally, it is ok not to know everything.

the marching orders emanate directly or indirectly from your office, and all the day-to-day decisions are made by you or by other people within the organization operating under your orders.

So, you have the power to tell your CFO to prepare a report for you, to authorize the payment of invoices (or to delegate up to a certain amount not exceeding your own authority) and to decide on the creation of a new line of products and so on. The articles of incorporation usually include some passage that states what kind of business the company is engaged in and might put limits on what kind of changes you are allowed to make without approval.

As a CEO you set the rules your company operates under and you determine who else has what kind of powers within the company. For instance, you could delegate some of your work to others that you feel are more qualified or that can help you lighten the load. Even so, you are still responsible for whatever it is that they do. Although it is great to work with others and to be able to trust them, the end responsibility is yours. You are still required to check up on them to make sure that they actually do what they say they do. Trust but verify is the mantra of being a CEO (and that includes this text).

As a CEO, you are allowed (within the boundaries set by the law) to fire other employees with cause. In some cases and jurisdictions, you can remove powers that you previously handed out.

So if you are not happy with the performance of an employee that was previously hired, then you have the power to terminate the relationship. This may cost the company (severance pay for instance, or if the employee feels they've been let go without proper cause, a wrongful termination suit), but it is definitely within your power to determine who works for the company and in what capacity.

What you are explicitly not allowed to do

You are *not* allowed to cause the company to break the law.

You are *not* allowed to enter into obligations that you know (or should have known) the company cannot fulfill.

You are *not* allowed to place your own interests above those of the company. Conflicts of interest should be avoided at all times.

Failure to observe these rules gets you into territory that is labeled 'improper conduct' and that is something you really want to avoid.

Some general observations

It is important to note that having a legal advisor that looks out just for you (and that you pay) is an absolute must. They will be invaluable when you enter into important obligations, when you are doubtful about the direction that others want you to steer the company in, or when you are simply unsure about some corporate governance detail.

It is much better to pay a lawyer a few hundred bucks (or euros or whatever currenty you use) and to become a little wiser than it is to assume that you can wing it and end up with a responsibility that you should not have taken on.

Note that legal advisors come in all shapes and sizes, and that there are many different specializations within the general profession of legal advisor (or lawyer, if you wish). For instance, there are people that specialize in employment law, intellectual property, tax law, corporate law and so on and depending on the situation, you may need one or more of these specialists in order to guide your decisions. Making uninformed decisions could easily result in you or the company breaking the law, so make sure that you *really* know what you are doing. Make sure you get your advice in writing, and that if it still does not sit right with you that you seek a second opinion.

A word on record keeping. As a CEO you will make decisions that can have far-reaching consequences. It is important, for yourself today and for your future self, in case there should ever be a discussion about your conduct as a CEO, to keep track of major decisions, when they were made and what went into the making of that decision.

If you have done a proper

job of documenting that you did what you could to keep the company on the straight and narrow, that you had sought (written) outside advice on areas that are not your expertise, that you made sure the books were kept properly, and that you made sure the company did not enter into any obligations it could not meet, you will sleep soundly and will have an easy time handing over the reins of the company to a successor if and when the time comes.

Failure to keep records, to communicate your orders verbally, to receive verbal input, and to base your decisions on that (such as promises to perform and other vague and ambiguous inputs), can lead to problems. If you are going to make decisions based on factors beyond your control, do what you can to nail these things down before committing to an irrevocable course of action.

Do not put your signature on a document that you haven't read or fully understood (including all its implications). Know your obligations, your rights and your personal limitations in terms of knowledge and expertise. Work with competent advisors (legal and otherwise) to fill any gaps to ensure that you are doing the right thing. Study up on this stuff and read your corporate documents (for instance, the articles of incorporation) in their entirety, and make sure you understand all of them — they may change your position in important ways.

If you are both a shareholder and a CEO, note that these are *two* different roles and that you cannot properly function if you are wearing two hats at the same time. In such situations, it may be useful to limit your perspective to one of your roles to make sure that you do not end up in a conflict of interest situation.

A note on ethics

Now, all of the above centers on legality, but of course there are other dimensions to running a company. It would definitely make the world a much better place if we didn't all focus so much on what is legal but also on what is *right*. In other words, be ethical.

If it is legal to pollute, or to externalize some kind of element that you would rather be rid of (knowing full well that it is legal for you to do so but which will cause misery somewhere else), then you may be on the right side of the law, but you do not amount to much in my book as a human being.

To some extent it seems that corporations are driven by people who will do everything they can within the letter of the law, while at the same time leaving the world as a whole a much worse place, and this deserves significant attention. Unfortunately, there is a major drive to achieve wealth (especially for shareholders) at any cost, and this is a systemic problem. There are CEOs out there who realize this and that makes their companies examples worth emulating.

Ray Anderson from Interface comes to mind as one example of a CEO who not only gets the legal bits right but who also steered a major corporation from being a huge destructive force into as clean a company as could be done with the technology of the day.

Goals are rarely in alignment

What is also important to remember is that it is very rare to have incentives for board members, founders, the CEO and shareholders/investors to be perfectly aligned. It is *impossible* to satisfy everybody's wishes, desires and demands at all times, and it is quite dangerous to assume that such an alignment is in effect when you ask for advice. What is good for the company or for you may not be the same thing as what is good for a specific shareholder, especially if that shareholder has a different approach to risk than you yourself are comfortable with.

Wrapping up

Finally, it is ok not to know everything. Nobody does, not even the best CEOs in the world know everything. But what they do know is what they *don't* know, and then they go and get educated or ask for (written) guidance.

Because I feel this is one of the most important articles I've written to date, I would very much appreciate it if you bring to my attention any points that you disagree with or if there is a factual error (jacques@mattheij.com). ■

Reprinted with permission of the original author. First appeared at jacquesmattheij.com.

Patrick J. McGinnis talks about being a 10% entrepreneur

Patrick J. McGinnis is a venture capitalist and private equity investor who founded Dirigo Advisors, after a decade on Wall Street, to provide strategic advice to investors, entrepreneurs, and fast growing businesses. He is the author of *The 10% Entrepreneur*, published by Penguin books.

Wish you had the bandwidth to launch a startup? Venture capitalist and author, Patrick J. McGinnis, says it's possible to start a business without quitting your day job.

Let's start from the beginning. You are a venture capitalist and writer based in New York. And you are a 10% entrepreneur. What exactly is a 10% entrepreneur?

A 10% entrepreneur is somebody who dedicates at least 10% of his or her time and, if possible, 10% of his or her money to invest, advise and start new ventures.

For computer professionals with a startup mindset, how can your book help?

You have a special and valuable skillset that many other people would love to tap into. At the same time, new ventures – such as startups may not have the cash required to hire top talent. You can trade your time, and do so in a flexible way, in exchange for an ownership stake in new ventures. And you don't have to leave your day job and take tremendous risk to do so.

You seem to advocate not quitting one's day job to pursue their entrepreneurial dreams. Is there ever an instance when you would advise someone to quit their job and jump right into building a startup?

I think it's always wise to prove your business model and validate that it can be successful before jumping in feet first. You might move a little slower, but you'll give yourself more time and more flexibility. Once you've gotten to that point and you can raise financing or you can self-fund, then the time may be right to quit your day job. Think very carefully before jumping in full-time and living off your savings to start a venture because that puts tremendous pressure on you and can make the whole enterprise far less sustainable.

For folks already building a startup, how can the 10% mindset help?

It’s a terrific way to bring on advisors and talent (like designers, marketing experts and lawyers) on a part-time basis since you can pay then with equity rather than with cash (which may be in short supply for an early-stage venture. You can also look for angel investors who are 10% entrepreneurs and who have specific skills to bring to the table. Finally, if you still have your day job and are starting a company, you can apply this approach to lessen your risks while you build your company.

You can apply this approach to lessen your risks while you build your company.

Recently, news about venture capital funding drying up has been worrying to a lot of entrepreneurs. What is your take on this?

Great startups will always find funding. Mediocre to poor startups will not.

We all know that despite the best laid plans, things can go wrong and startups fail. What is your advice to someone who has gone down the startup road and turned up empty?

I also recommend that full-time entrepreneurs think like 10% entrepreneurs. I call these people 110% Entrepreneurs. When you’re running a startup, you’re placing a huge bet – a bet that will determine your future wealth and success on a high risk venture. Why not create some diversification by using all of the skills you’re developing as a full-time entrepreneur to also engage with other startups as an advisor or even an angel?

In your book you profile real-life entrepreneurs who have cracked the secret to making it big. What would you say is the common quality these folks share that helped them succeed?

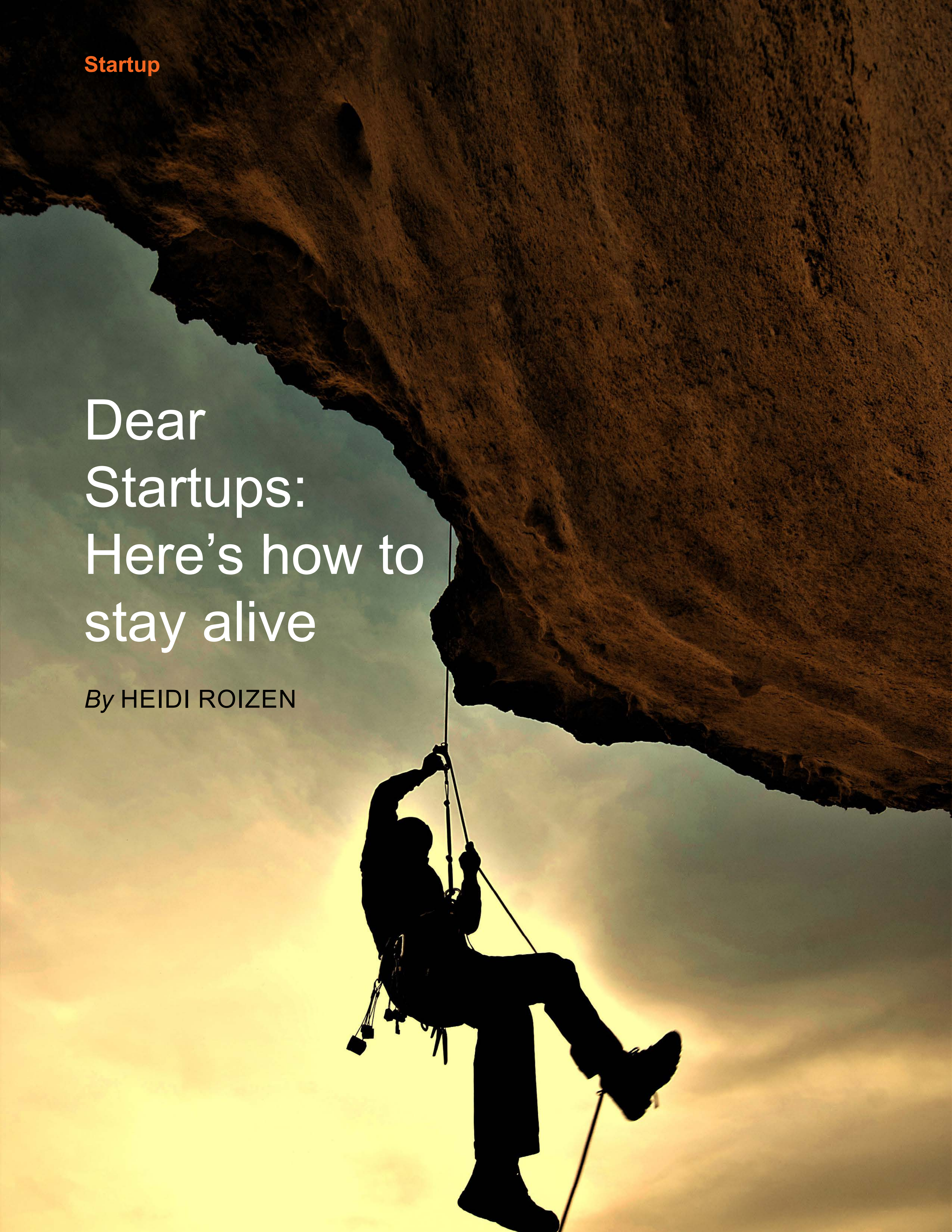
I think that each of these people defined success for themselves. Being a part-time entrepreneur is a more pragmatic approach to building a business. Some people will question why you don’t jump in 100%. Others will question why you’re working so hard. But for each of the people in the book, the venture they chose to pursue mapped closely with their skills, interests, and definition of success. Because of that they enjoyed the journey and were willing to stick it out through the natural ups and downs of entrepreneurship.

Where can people find out more about you? And where can they pick up a copy of your book?

You can order from patrickmcginnis.com or at any major bookseller. You can also find more information on my website and on Twitter @pjmcginnis. ■

Interview was edited for brevity and clarity.

Startup



Dear Startups: Here’s how to stay alive

By HEIDI ROIZEN

There are storm clouds gathering over Silicon Valley — and it's more than just El Niño.

As a venture capitalist, I see a lot of data points within the private company marketplace. Every Monday, I sit in a room with my partners and we discuss dozens of companies, both portfolio companies as well as those we are considering for investment. When a market turns, we tend to see the signs earlier than the entrepreneurs working on the front lines.

This market? I'd say it has turned.

It is going to be hard (or impossible) for many of today's startups to raise funds. And I think it will get worse before it gets better. But, hey, my entrepreneurial friend, who ever said it was going to be easy? One of my favorite expressions is: "that which does not kill us makes us stronger."

This market? I'd say it has turned.

So which is it going to be for you? Tougher? Or dead.

Fortunately (*unfortunately?*), I've been to this movie before, during the dot-com "nuclear winter" — anyone remember that? I'd like to think I've learned some things from that painful experience.

I've seen companies live, and I've seen them die. And I've concluded that certain behaviors separate the two.

Which behaviors, you ask? *Here are a few from my downturn playbook for how to stay alive.*

Stop clinging to your (or anyone else's) valuation

You know what somebody else's fundraise metrics are to you? Irrelevant. You know what your own last round post was? Irrelevant. Yes, I know, not legally, because of those pesky rights and preferences. But emotionally, trust me, it is irrelevant now. We even have a name for this – *valuation nostalgia*. Yes, it was great when companies could raise those amounts, at those prices, *blah, blah, blah*, but the cheap-money-for-no-dilution thing is largely over now. The sooner you get on with dealing with that, and not clinging to the past, the better off you will be. As my DFJ partner Josh Stein says, "flat is the new up."

Redefine what success looks like

I had lunch last week with a friend of mine who broke her leg in three places four months ago. "I used to think a successful weekend was 10+ miles of running," she said. "For now, success is going to have to mean making it to the mailbox and back without my crutches." When a market like this turns, in order to survive, it is critical to redefine what success is going to look like for you — and your employees, investors, and other stakeholders. Holding on to 'old' ideas about IPO dates, large exits and massive new up rounds can ultimately be demotivating to your team. If you can make it through the downturn, you will have those opportunities again. But for now, reset your goals.

Get to cash-flow positive on the capital you already have (aka, *survive*)

My DFJ partner Emily Melton said this in our last partner meeting: "Must be present to win." I used to say it at T/Maker (the company for which I was CEO) in a slightly different way: "In order to have a bright long-term future, we need to have a series of survivable short term futures." You need to survive in order to ultimately win.

You know what kind of companies generally survive? *Companies that make more money than they spend.* I know, *duh*, right? If you make more than you spend, you get to stay alive for a long time. If you don't, you have to get money from someone else to keep going. And, as I just said, that's going to be way harder now. I'm embarrassed writing this because it is so flipping simple, yet it is amazing to me how many entrepreneurs are still talking about their plans *to the next round.* What if there is no next round? Don't you still want to survive?

Yes, some companies are 'moon shots' (DFJ has a fair number of those in our portfolio) where this is simply not possible. But for the vast majority of startups, this *should* be possible.

So, for those of you in the latter group, I want you to sharpen your spreadsheet, right now, and see if, by any hugely painful series of actions, you could actually be a company that makes money. ASAP. Or at least before you run out of money. Because that's the only way I know to control your own destiny. You don't have to act on it (although I would), but at least you will know if you have a choice.

And if you absolutely, positively, cannot get there without more capital? Then you need to…

Understand whether your current investors are going to get you there

Guess who else cares about whether you live or die? Yep, your current investors. Another duh. That's why they are your best source of 'get me to cash flow positive' financing. And yet, even though we all know this, why is it we don't actually (1) create the plan that gets us to cash flow positive ASAP, and then (2) go to our backers and get their commitment that they will see us through (or know that they won't, because if they won't, the sooner we know that, the sooner we can go out and do something about this.) I know many VCs hate to be put on the spot about this, but I think entrepreneurs have the right to ask, and to know.

Stop worrying about morale

Yes, you heard me right. I can't tell you how many board meetings I've been in where the CEO is anguished over the impacts on morale that cost cutting or layoffs will bring about.

You know what hurts morale even more than cost-cutting and layoffs? *Going out of business.*

I was at a conference once where someone asked Billy Beane how he created great morale at the A's. His answer? "I win. When we win, morale is good. When we lose, morale is bad."

Your employees are smart. They know we are in uncertain times. They see the stress on your face. They worry about their jobs. What do they want to see most? A decisive plan for survival, that's what — even if some of them have to go. Trust me, a clear plan is a real morale turn-on.

Cut more than you think is needed

Yes, this is simple, but not easy. It is so easy to justify why you want to lay off fewer people. However, when you do, by and large, you'll be laying even more off later. Why we humans seem to prefer death by a thousand cuts is a mystery to me. Don't. It's easier on everyone if you cut deeper and then give people clarity about the stability of the remaining bunch.

In order to have a bright long-term future, we need to have a series of survivable short term futures.

> ***These markets generally take a long time to recover. Longer than you think.***

Scrub your revenues

Last week, an entrepreneur pitched us, and his ‘current customers’ slide was alight with bright, trendy logos of bright, trendy venture-backed companies.

You know what I saw? A slide full of *bright, trendy, money-losing, may-not-survive companies.* (Luckily, in this case, the entrepreneur referenced these customers because he thought VCs would like to see that their smart startups use his stuff, but he actually had a lot of mainstream customers, too. He has a new slide now.)

This, I think, was one of the biggest surprises from the last dot-com bust — we all knew we had to cut our expenses, but no one thought about what our customers might be doing.

And guess what? They were all cutting costs, too — including those costs which comprised our revenues. *Or worse, they were going out of business.*

If you are in Silicon Valley and your customers are mostly well-paid consumers with no free time, or other venture-backed startups, well, I’d be worried. And yes, it sucks, but it is better to be worried than surprised.

Focus maniacally on your metrics

I know a few CEOs who delegate the understanding of their financials and their business metrics to the CFO, and then stop worrying about all that ‘numbers stuff’.

Don’t do that. You have to know your numbers inside and out — they are your life blood. You also have to know which metrics drive the business, and focus on them like your survival depends on it — because it does.

Figure out your canary (or canaries) in the coal mine (by that I mean the leading indicators that tell where your business is headed and whether it is healthy) and watch them weekly, daily, in real time or whatever is possible. And, have a plan in advance about what you will do if/when the metrics go south.

Many of the best companies to have survived the last downturn became super data-driven, and were constantly course-correcting to make small but continuous improvements in their operations with what they learned.

Hunker down

These markets generally take a long time to recover. Longer than you think. And, it might get worse. So don’t plan for the sun to start shining tomorrow. Or next month. Or next quarter. Or maybe even next year. Sorry.

Having just thoroughly depressed you, let me say that I’ve seen amazing transformations by companies who adapt early to the new reality. Severe budgets give clarity. Smaller teams often find greater purpose in their work. Gaining control (by becoming profitable) feels really, really good. Watching your competition (who didn’t read this) die, feels — can I say it? — well let’s just say that when your competition goes out of business, you often gain their customers…and that’s a very, very good thing.

Some of the greatest companies were forged in the worst of times. May you be one of them. ■

Reprinted with permission of the original author. First appeared at heidiroizen.tumblr.com.

How I sold my Bible app company

By TREVOR MCKENDRICK

It all started with an email I wrote last November:

> *I'm a Spanish Bible salesman. Kind of.*
>
> *The goal was just to pay our rent, but instead the Spanish Bible app I made turned into a successful 6-figure business.*

That was sent to Gimlet Media November 12, 2014 pitching my Bible app story for their podcast, Startup.

12 days later I was being interviewed by Alex Blumberg.

Gimlet Media published that episode on January 5. My interview ended up being half of the episode and I thought it went great. I'm a huge *This American Life* and *Startup* fan so just being interviewed by Alex was fun, let alone being on the show.

I got a few "congrats!" on Twitter/Facebook etc. and I thought that was the end of it.

Until *Business Insider* found it.

The first email

About a month later in February, *Business Insider* published a post that summarized the interview.

This would end up changing my life.

Business Insider's reach is incredible, and as a result of that post:

- The story was written about on over 1,000 other websites
- It was shared over 4,000 times from Fox News' Facebook page
- They discussed it multiple times on the the Fox News morning show
- I was a guest on the *Huffington Post's* morning show
- I was interviewed by a bunch of other smaller media sites

And of course the kicker: the following email landed in my inbox on February 6th:

> *Trevor,*
> *Great to meet you. I work for Salem Media and I read the story about your Bible App. Sounds like you have really created something that is doing very well. I would like to talk with you because* ***we are interested in buying the app from you if you have any interest****. You can check out me on LinkedIn if you want and we can set up a quick call just to say hello and see if there is anything there...*

(Emphasis mine)

I instantly recognized the company and knew this was a very real offer.

Salem Media

Back in 2012 when my apps started really picking up steam I dug into and learned a ton about the Christian media industry.

(For example, I learned that the #1 selling English Bible, the NIV, is owned by Rupert Murdoch and News Corp.)

In my research I found a public company called Salem Media, worth $160 million as I write this. They got started in the 80's as a radio company.

I pored through their 10-K/10-Q filings, in particular their acquisition sections.

I found things like:

> *"On May 15, 2012, we purchased Churchangel.com and rchurch.com for $0.2 million. "*

> *"On August 30, 2012, we acquired SermonSpice.com for $3.0 million."*

> *"On October 1, 2012, we completed the acquisition of Godvine.com for $4.2 million."*

The joy of doing business with public companies; all this info about previous acquisitions, prices, dates... available to anyone.

Even back in 2012 it was apparent very quickly that they buy digital properties all the time.

Websites, apps, Facebook pages, domains... you name it.

Back in 2012 I had no way of getting their interest. But when that first email from them landed in my inbox in 2015, I knew it was real instantly, and that it was game on.

I didn't respond until 24 hours later. The negotiations had already begun.

The whole process ended up taking 7 months, and basically played out as follow:

1. Agree on price
2. Due diligence
3. Asset Purchase Agreement term negotiations
4. Closing

1. Agree on price

After we each signed an NDA we got on the phone. My contact for this entire process was a nice guy named David. It's his job to be nice and get good terms from target companies. He does his job well.

David walked me through basically their entire process, e.g. they make acquisitions of all shapes and sizes, they like what I'm doing, would I be interested in hearing an offer, etc.?

I said that all sounded fine, so he sent over an email with some preliminary questions:

1. What analytics do you use to monitor the app? Is this something you can give me read only access too?
2. Can you give me the revenue numbers by month for the last 12 months?
3. What are the monthly sessions for the last 12 months?
4. What are the monthly active user numbers for the last 12 months?
5. What are the monthly download numbers for the last 12 months?
6. Do you collect email addresses as part of your business?
7. Are there any contracts, litigation or business reasons that would prevent you from being able to sell the app to us?

Pretty reasonable stuff, so I sent over a detailed email with answers, numbers, etc.

And then I didn't hear anything for a week.

Thankfully I'd also received interest from a different, private investor.

I was nursing both these leads simultaneously and it was around this exact time that I got an official offer from the private investor. It was too low but that didn't matter. I could still use it to negotiate with Salem Media.

It was a late Tuesday night for him on the east coast, but after a week of silence I got a response to my "I have another offer email" just 15 minutes after I hit send:

> *I would like to move forward I am at a conference all week. Can we talk Monday? Sorry for the delay.*

A few more emails/phone calls later (this became a theme) they finally made their first offer. It was around 3.5x revenue. I said thank you and that I'd get back to him.

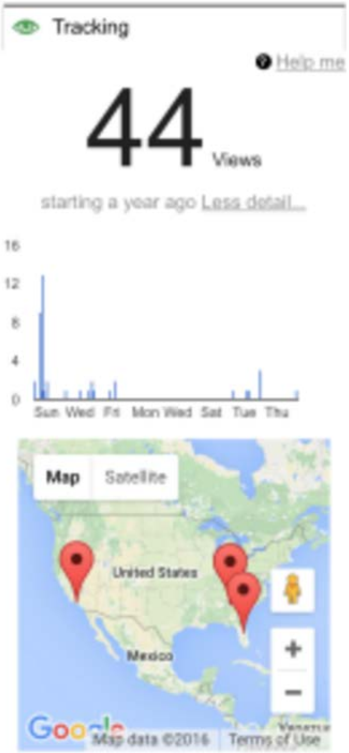
I thought about it, and here's what I knew:

- They were already giving the apps a relatively high valuation
- I knew their first offer wasn't their best
- I didn't have to sell because the apps made easy money by themselves
- If anyone was going to pay top dollar for my apps, it was these guys. As they say, the time to eat the hors d'oeuvres is when they're being passed.

David had mentioned to me that most of his team worked remotely. For example, he's based on the east coast, but his boss and Salem board member (who had authority to yay/nay my deal) worked at headquarters out of their offices in California.

I also used a CRM called Streak that, among other things, tells you where people are when they read your email.

During the times when I was waiting for them to get back to me I had no idea what was going on on their end. Were they really that interested? Were they just busy? Were they talking about it?



▲ Joy of Streak tracking

The joy of Streak, and the fact that they work remotely, is that I could *see them forwarding my email to each other*. You'll see in the map below that it was opened in multiple locations, multiple times.

It's like sending a text to someone you're interested in after a date, and knowing that they're talking to their friends about how to respond. They're interested.

This doesn't always work obviously, but it gave me a ton of confidence in the moment.

My goal became to convince them that the apps were a strategic purchase (vs. just straight cash flow).

I responded with the following email:

> *David,*
> *I’m still interested in moving forward and appreciate the offer.*
>
> *The offer was enough to begin a serious conversation, but I think it was on the low side. Here’s why:*
>
> *– We’re the #1 downloaded Spanish-language Bible in the App Store. In fact, we get more downloads than the vast majority of English bible apps.*
>
> *– We’re a Top 10 Free app in the Books category. Better than Disney, Google’s Play app, Barnes & Noble, Marvel Comics, DC Comics, and others.*
>
> *– The business is very consistent, and has been for some time (> 2.5 years). This isn’t a trend or a seasonal app.*
>
> *– We rank #1 for “la biblia”, #2 for “biblia”, #2 for “spanish bible” and top 10 for many other phrases.*
>
> *Since the app performs so consistently and takes so little of my time (~1 hour per month) it needs to make financial sense to give it up…*
>
> *Given the strategic value of the app, Salem Media’s unrivaled ability to monetize media properties, and the other factors mentioned above, I could accept an offer of 6x revenue. That would give Salem Media the most popular Spanish-language Bible app at a reasonable price and provide me enough incentive to give it up.*
>
> *Frankly it also helps that you guys have experience with app properties and that I can trust you as a buyer.*
>
> *Let me know what you think. I can move quickly if the deal makes sense.*
>
> *Trevor*

They came back with an updated offer of 4.5x revenue, a bump of 1x.

But I knew they could do more.

I countered again, basically reiterating the previous email, lowering my ask to roughly 5x revenue.

Finally, he emails me:

> *Trevor,*
>
> *Let me know if you have a few minutes this afternoon. I think I have a way to get to 5x revenue but I want to be sure you are good with it before I send along the offer.*
>
> *Hope you all are having a great trip.*
>
> *David*

The timing for this entire process was less than ideal as my wife and I were in Europe for all of March and April.

But I still remember where I was when I “accepted” their offer: having dinner with my wife in Germany.

I’m not gonna lie, saying yes on the phone from a restaurant in Germany to a big company offering to buy my company felt pretty cool.

Once we’d agreed on the price it was time to begin due diligence.

2. Due diligence

After we’d agreed on price, the joy of due diligence began.

I already had all my ducks in a row so it was pretty easy to get them everything they asked for (see *Aquisition Questions List* on next page).

What wasn’t fun was how long it took. We agreed on the price and started due diligence around March 10th, and I didn’t get the first draft of the initial asset purchase agreement until April 21st. Just awful.

In the first half of April they’d asked (and I’d complied with) multiple due diligence asks, phone calls, etc. Eventually I had to put my foot down and say no more. I wasn’t giving them a single more minute or document until I got the draft agreement.

It worked.

I worked up more and more moxy as the process continued.

3. Asset Purchase Agreement (APA) term negotiations

I finally got the draft APA on April 21st. It was a 63-page document, so I finally hired an M&A firm to help.

I ended up only having to spend ~$5k on lawyers for the whole deal, which sounds like a lot until you work with lawyers. I was happy to pay it for the advice I got. I actually wish I'd hired them a little bit sooner.

There were quite a few things we negotiated to change in the APA, but the two that took the longest were:

- Confidentiality
- Indemnification

Confidentiality

Basically I spent 2 weeks convincing them to let me write this blog post!

No joke. The original terms were something along the lines of that I couldn't tell anyone about the sale, ever, which is crazy since it's going to be publicly available in their Q3 10-Q in about a month anyway.

Thankfully they finally agreed to let me talk openly about most things in the transaction, which is why I'm able to write this post.

Indemnification

For people who aren't familiar: indemnification means "if I get sued for something you did, will you protect me?"

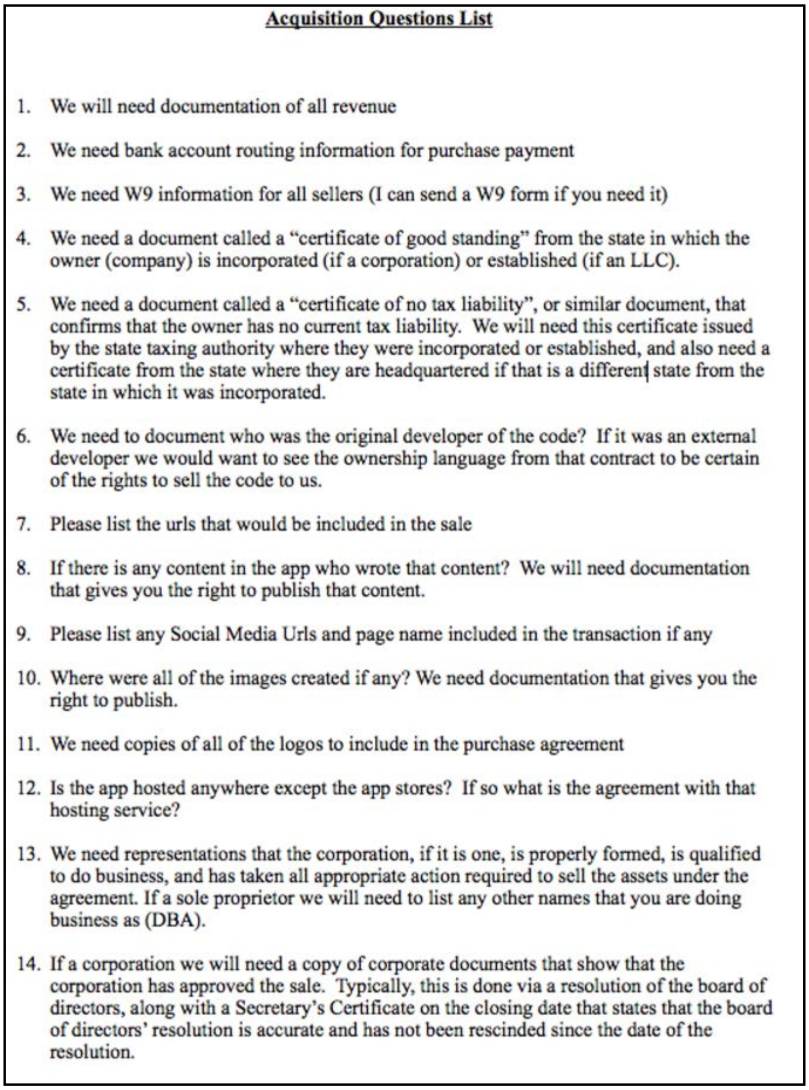
This is generally a reasonable term and is included in basically every M&A deal under the sun.

Generally the pieces that get negotiated are 1. how long does it last and 2. whether there's a liability cap.

E.g. How long after the sale do I have to indemnify them, and in some disaster scenario if they got sued and lost, what's the max I'd ever owe them?

In the end we agreed on a length of time and cap that was comfortable for both of us.

I will say this: I had to push back hard to get my terms on this point. Initially they wanted *uncapped liability forever*, which was unacceptable.



Acquisition Questions List

1. We will need documentation of all revenue
2. We need bank account routing information for purchase payment
3. We need W9 information for all sellers (I can send a W9 form if you need it)
4. We need a document called a "certificate of good standing" from the state in which the owner (company) is incorporated (if a corporation) or established (if an LLC).
5. We need a document called a "certificate of no tax liability", or similar document, that confirms that the owner has no current tax liability. We will need this certificate issued by the state taxing authority where they were incorporated or established, and also need a certificate from the state where they are headquartered if that is a different state from the state in which it was incorporated.
6. We need to document who was the original developer of the code? If it was an external developer we would want to see the ownership language from that contract to be certain of the rights to sell the code to us.
7. Please list the urls that would be included in the sale
8. If there is any content in the app who wrote that content? We will need documentation that gives you the right to publish that content.
9. Please list any Social Media Urls and page name included in the transaction if any
10. Where were all of the images created if any? We need documentation that gives you the right to publish.
11. We need copies of all of the logos to include in the purchase agreement
12. Is the app hosted anywhere except the app stores? If so what is the agreement with that hosting service?
13. We need representations that the corporation, if it is one, is properly formed, is qualified to do business, and has taken all appropriate action required to sell the assets under the agreement. If a sole proprietor we will need to list any other names that you are doing business as (DBA).
14. If a corporation we will need a copy of corporate documents that show that the corporation has approved the sale. Typically, this is done via a resolution of the board of directors, along with a Secretary's Certificate on the closing date that states that the board of directors' resolution is accurate and has not been rescinded since the date of the resolution.

The most tense moments in the entire 7-month process was on a phone call about this very subject.

I told them I was going to walk away from the deal (after ~6 months of work) if I didn't get some sort of compromise. That was followed by roughly 45 seconds of silence on the other side of the line, but eventually the guy spoke up and said he'd see what he could do.

And I wasn't bluffing.

From this phone call we were able to finally agree on indemnification terms that were acceptable to both parties.

4. Closing

The best day of all.

Once we'd agreed on every detail in the APA it was time to schedule a day to close the deal. In my case that meant signing and transferring the assets all in the same day (in high profile acquisitions you see online the announcements happen after the deal is signed but usually before it's closed.)

▲ My "celebration" at In-N-Out

So sure enough: I transferred the apps from my iTunes Connect account to theirs. I gave them my login details for my email list, my hosting account, and a few other things.

Then they wired over the money.

And just like that the deal was over and I'd sold my first company.

My wife was out of town so to "celebrate" I treated myself to In-N-Out for lunch!

Mental exhaustion

The deal made a ton of sense and I'm glad it got done.

But it did take over 7 months, hundreds of emails, and dozens of phone calls. It was psychologically exhausting and definitely the hardest part of the 3.5 years I ran the business.

I'd heard so many horror stories about failed acquisitions that I always assumed it was dead until the cash was finally in the bank.

Even literally the Monday before the sale closed I spent all day in bed worried about whether it would actually go through. That was not fun, and various levels of depression occurred throughout the sales process.

That would explain the relieved smile in the picture above.

Thank you Salem Media. It was a pleasure doing business with you.

Here's to the next side project. ■

Reprinted with permission of the original author. First appeared at trevormckendrick.com.

tcpdump is amazing

By JULIA EVANS

It took me 2 years, but I think now I love tcpdump. Before we go into why — what's tcpdump?

tcpdump is a tool that will tell you about network traffic on your machine. I was scared of it for a long time and refused to learn how to use it. Now I am wiser and I am here to show you that tcpdump is awesome and there is no need to be scared of it. Let's go!

tcpdump: the basics (or: how not to use it)

If I just run `sudo tcpdump -i wlan0` (listen to wireless network traffic plz!), tcpdump says this:

```
23:48:26.679315 IP 206.126.112.170.https > kiwi.
lan.47121: Flags [P.],
 seq 1:42, ack 2294, win 1672, options [nop,nop,TS
val 675931991
  ecr 60685517], length 41
```

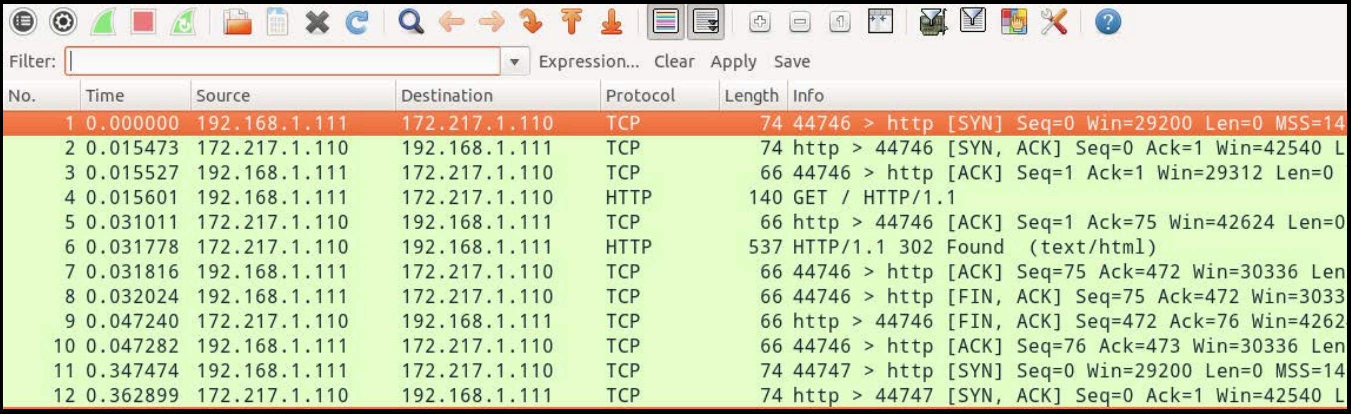
The first time I ran tcpdump I took one look at some output like that, went WELP NOPE NOPE NOPE NOPE NOPE NOPE and gave up on tcpdump. what is an ecr? a win? flags? oh god.

I don't know what hardly any of this means (though, I wrote a tiny TCP stack one time so I sorta know. But not enough to help too much.)

So, we've learned that we need to pass some... options... to tcpdump to actually make use of it without being a TCP wizard. But what options? We'll find out! First, let's get concrete about the problems we're trying to solve.

The case of the slow HTTP request

Let's suppose you have some slow HTTP requests



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.1.111	172.217.1.110	TCP	74	44746 > http [SYN] Seq=0 Win=29200 Len=0 MSS=14
2	0.015473	172.217.1.110	192.168.1.111	TCP	74	http > 44746 [SYN, ACK] Seq=0 Ack=1 Win=42540 L
3	0.015527	192.168.1.111	172.217.1.110	TCP	66	44746 > http [ACK] Seq=1 Ack=1 Win=29312 Len=0
4	0.015601	192.168.1.111	172.217.1.110	HTTP	140	GET / HTTP/1.1
5	0.031011	172.217.1.110	192.168.1.111	TCP	66	http > 44746 [ACK] Seq=1 Ack=75 Win=42624 Len=0
6	0.031778	172.217.1.110	192.168.1.111	HTTP	537	HTTP/1.1 302 Found (text/html)
7	0.031816	192.168.1.111	172.217.1.110	TCP	66	44746 > http [ACK] Seq=75 Ack=472 Win=30336 Len
8	0.032024	192.168.1.111	172.217.1.110	TCP	66	44746 > http [FIN, ACK] Seq=75 Ack=472 Win=3033
9	0.047240	172.217.1.110	192.168.1.111	TCP	66	http > 44746 [FIN, ACK] Seq=472 Ack=76 Win=4262
10	0.047282	192.168.1.111	172.217.1.110	TCP	66	44746 > http [ACK] Seq=76 Ack=473 Win=30336 Len
11	0.347474	192.168.1.111	172.217.1.110	TCP	74	44747 > http [SYN] Seq=0 Win=29200 Len=0 MSS=14
12	0.362899	172.217.1.110	192.168.1.111	TCP	74	http > 44747 [SYN, ACK] Seq=0 Ack=1 Win=42540 L

▲ **Figure 1:** Wireshark

happening on your machine, and you want to get a distribution of how slow they are. You *could* add some monitoring somewhere inside your program. Or! You could use tcpdump. Here's how that works!

1. Use tcpdump to record network traffic on the machine for 10 minutes
2. Analyze the recording with Wireshark
3. Be a wizard

The secret here is that we can use tcpdump to record network traffic, and then use a tool that we're less scared of (Wireshark) to analyze it on our laptop after.

Let's do it! Let's say I want to record all TCP traffic to port 80 (so, HTTP). Then I can record traffic with:

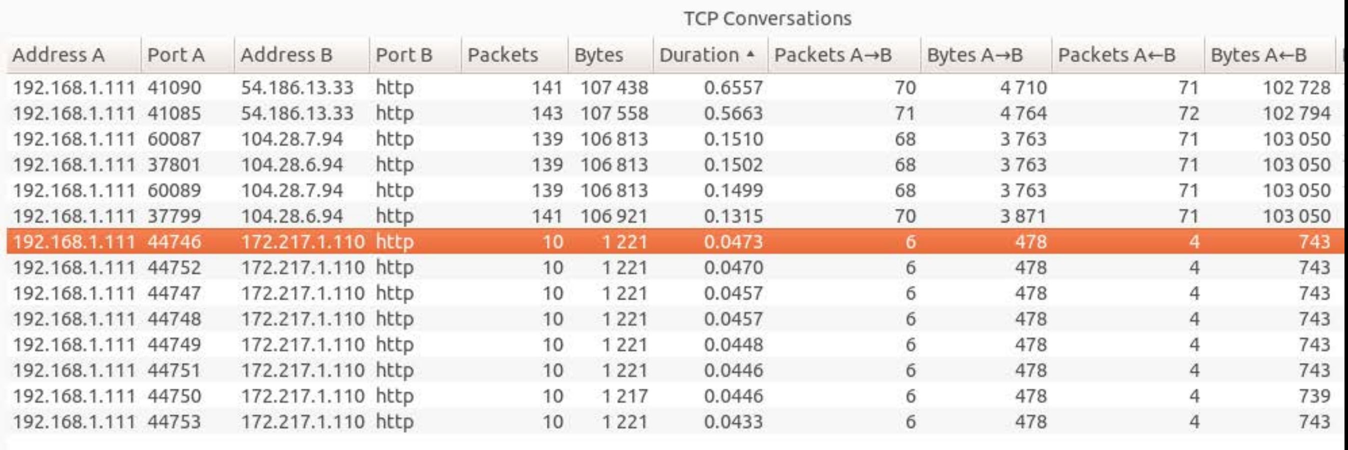
```
$ sudo tcpdump -i wlan0  \
               src port 80 or dst port 80 \
               -w port-80-recording.pcap
```

This filters for only packets to or from port 80 (the name for this syntax is "pcap filters" and they are THE BEST) and saves a recording to `port-80-recording.pcap`.

Next up, Wireshark! I'm going to start it with `wireshark port-80-recording.pcap`. Here's what we see to start. (see Figure 1)

That's a little intimidating. Every time I make a HTTP request that might be 200 TCP packets, which are a huge pain to recognize and make sense of by hand. But we can fix it! I clicked on Statistics -> Conversations, where it organizes all these disparate packets into TCP sessions. Let's see what that looks like! (see Figure 2)

This is already a lot more understandable,



TCP Conversations

Address A	Port A	Address B	Port B	Packets	Bytes	Duration ▲	Packets A→B	Bytes A→B	Packets A←B	Bytes A←B
192.168.1.111	41090	54.186.13.33	http	141	107 438	0.6557	70	4 710	71	102 728
192.168.1.111	41085	54.186.13.33	http	143	107 558	0.5663	71	4 764	72	102 794
192.168.1.111	60087	104.28.7.94	http	139	106 813	0.1510	68	3 763	71	103 050
192.168.1.111	37801	104.28.6.94	http	139	106 813	0.1502	68	3 763	71	103 050
192.168.1.111	60089	104.28.7.94	http	139	106 813	0.1499	68	3 763	71	103 050
192.168.1.111	37799	104.28.6.94	http	141	106 921	0.1315	70	3 871	71	103 050
192.168.1.111	44746	172.217.1.110	http	10	1 221	0.0473	6	478	4	743
192.168.1.111	44752	172.217.1.110	http	10	1 221	0.0470	6	478	4	743
192.168.1.111	44747	172.217.1.110	http	10	1 221	0.0457	6	478	4	743
192.168.1.111	44748	172.217.1.110	http	10	1 221	0.0457	6	478	4	743
192.168.1.111	44749	172.217.1.110	http	10	1 221	0.0448	6	478	4	743
192.168.1.111	44751	172.217.1.110	http	10	1 221	0.0446	6	478	4	743
192.168.1.111	44750	172.217.1.110	http	10	1 217	0.0446	6	478	4	739
192.168.1.111	44753	172.217.1.110	http	10	1 221	0.0433	6	478	4	743

▲ **Figure 2:** TCP sessions

If you have questions about network traffic on your machines, maybe tcpdump is the tool for you!

to me! There were 12 or so HTTP requests that happened. There's a 'Duration' column that tells me the total duration of the TCP session. So some of my requests took 47ms, and some of them took 655ms. The 47ms ones are Google, and the 655ms one is ask.metafilter.com. What's up, Metafilter? Who knows. Metafilter was sending me way more packets (google was just like "lol redirect", 10 packets, done), so I get that it takes more time. No big deal. That was super easy!

I did this at work recently because my metrics were reporting that some HTTP requests were taking like 100ms each. I ran tcpdump, did the Wireshark thing above, and Wireshark was like "yeah those are all taking 3ms. Your metrics are wrong, or at least counting something other than just the network request!". This was a very helpful fact to know.

With tcpdump I feel really confident that it's telling me the truth about what my network traffic is up to, because that's literally its whole job. And I can just capture packets and use it with Wireshark which is a really friendly and delightful tool.

pcap files

I mentioned really briefly that tcpdump lets you save pcap files. This is awesome because literally every network analysis tool in the universe understands pcap files. pcap files are like freshly baked chocolate chip cookies. Everybody loves them.

Filtering packets

Okay, so now let's imagine we're on a box where a lot is going on. We want to capture some TCP traffic to analyze it later. But not all the traffic! Only some of it. I mentioned before that you use "pcap filter rules" to do this. I only know how to do literally 2 things — filtering on port and IP address. Here's the 3 second Julia tutorial on pcap filter rules.

```
stuff being sent to port 80:
    dst port 80
you can use booleans!
    src port 80 or dst port 80
here's how to filter on IP:
    ip src 66.66.66.66
```

I don't know why it's `src port $ip` but `ip src $ip`. If I get it wrong I just try to switch the order. You can go read the docs and do much more complicated filtering but this has been good enough for me so far.

To learn more about this, read about the [Berke].

Overhead (is it safe to run tcpdump on my production machine?)

Short answer: I think so, mostly.

Longer answer: I don't quite know. But here's what I do know.

I watched this great talk by Dick Sites, who works at Google (which you should totally watch if you're into awesome performance stories) where he mentioned that any time he introduces a performance monitoring tool that takes up more than 1% of resources in overhead, he needs to have a long serious conversation with the datacenter administrators. He said that tcpdump is an example of something that's too expensive.

But I don't think his requirements are my requirements (if a thing I administer gets 5% slower for 10 minutes while I collect network packets, it's no big deal).

tcpdump uses this pcap filter language, and thomas ptacek mentioned to me on Twitter the other day that those filter rules are compiled down to something super efficient (with an optimizing compiler?).

My impression is your filter rules are collecting 500KB/s or something of network traffic, it's probably no big deal and you can go nuts with tcpdump on your production machines. If you're Netflix and you're trying to save 200MB/s of packets to disk, probably you will have a bad time? I don't know. I've never observed any bad effects from using tcpdump, but I do look `dstat` to get a sense for how much network traffic I might be capturing first, and try to filter appropriately.

Even more awesomeness: tshark can look inside your packets

So, now we know how to filter by IP and stuff, and use wireshark. Next, I want to tell you about `tshark`, which is a command line tool that comes with Wireshark.

tcpdump doesn't know about HTTP or other network protocols. It knows pretty much everything about TCP but it doesn't care what you put inside your TCP packets. tshark knows all about what's inside your TCP packets, though!

Let's say I wanted to spy on all GET requests happening on my machine. That's super easy with tshark:

```
$ sudo tshark -i any \
            -Y 'http.request.method == "GET"' \
            -T fields \
            -e http.request.method -e http.request.
uri -e ip.dst
GET   /hello.html     54.186.13.33
GET   /awesome.html   172.217.3.131
GET   /               172.217.3.131
```

This filters for just packets which have a HTTP GET request in them, and then prints out the request method and the URI that we're requesting for each one. It's beautiful! I had no idea this was even possible before. But it gets better! HTTP is pretty easy. Everyone knows HTTP. But tshark doesn't just know HTTP; it knows like EVERYTHING. Everything that Wireshark knows.

Yesterday at work, I wanted to know which Mongo collections were being queried from a specific machine. This was totally impossible with the tools I had. But nothing is impossible with tcpdump/tshark! It's just network traffic, after all. So I ran something like this:

```
sudo tshark -i any \
            -f src port $mongo_port or dst port
$mongo_port \
            -T fields \
            -e ip.dst -e mongo.full_collection_name
```

and since tshark totally understands the Mongo protocol, it immediately started printing out Mongo collection names, and I could see exactly what was going on. It was amazing. I'm super excited to use tshark more now.

Go forth and tcpdump

If you have questions about network traffic on your machines, maybe tcpdump is the tool for you! If you have cool tcpdump stories or other ways to use it that I haven't mentioned here, tell me on Twitter!

Also if you understand how to reason about the overhead of using tcpdump ("below 2 MB/s is always ok"?), I would REALLY REALLY LOVE TO KNOW. Please tell me. ■

Reprinted with permission of the original author. First appeared at jvns.ca.

Graphing when your Facebook friends are awake

By ALEXANDER HOGUE

Look I'm not really sure why but I think I made a thing that makes graphs of when people are online on Facebook. It sounds kinda creepy and uh, it is. Read along so you, too, can be the NSA.

Little green dots

You know those green dots on the sidebar on Facebook that tell you who's online? How do they get there? Also there are times next to people who are offline.

What are those about? (see Figure 1)

I was wondering the same things, and so one day I decided to 360 noscope hack Facebook by right clicking and selecting “Inspect Element.” (see Figure 2)

I’m in

We did it team. Anyway alright, uhhhh, let’s just, uh, snoop around here reallllll sneaky-like.

If you reload the page you’ll see approximately fifty-bajillion network requests go off as Facebook desperately tries to load all the junk that it needs to display facebook.com.

You might be wondering at this point why I decided to look for interesting things in this mess instead of, I dunno, getting out more, getting a cat, that sorta thing. Anyway hey look a heading.

Finding the good stuff

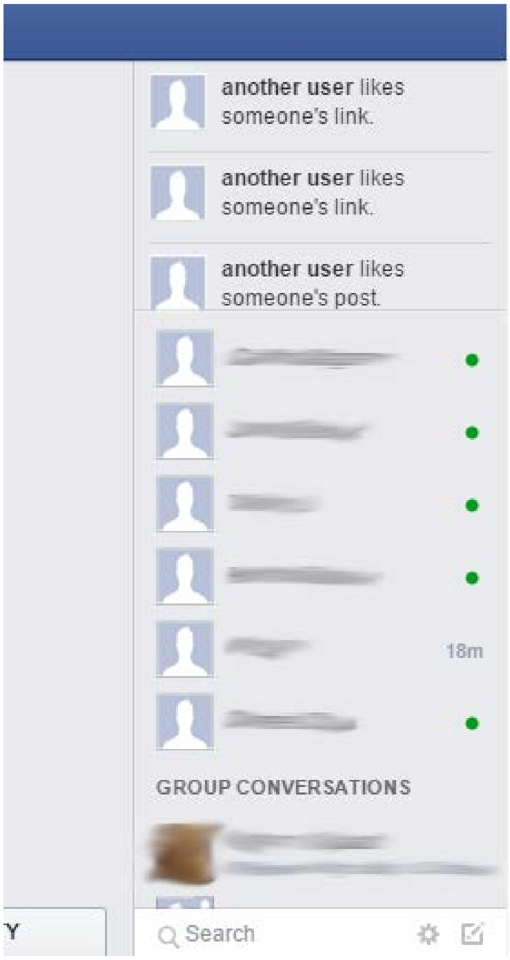
What’s this “pull” thing? (see Figure 3)

THAT looks like some #datascience right there. This is the kind of *100% legit secret undocumented “API”* that we came here for. Let’s do some reverse engineering.

It looks like a mapping of Facebook user ids to... their online status? But there’s more than one value? “webStatus” and “fbAppStatus” are both there. What’s more, it tells you what the person is doing on each of the different kinds of statuses.

For example:

- “messengerStatus” : “invisible” means they’re not online on the Facebook Messenger app.

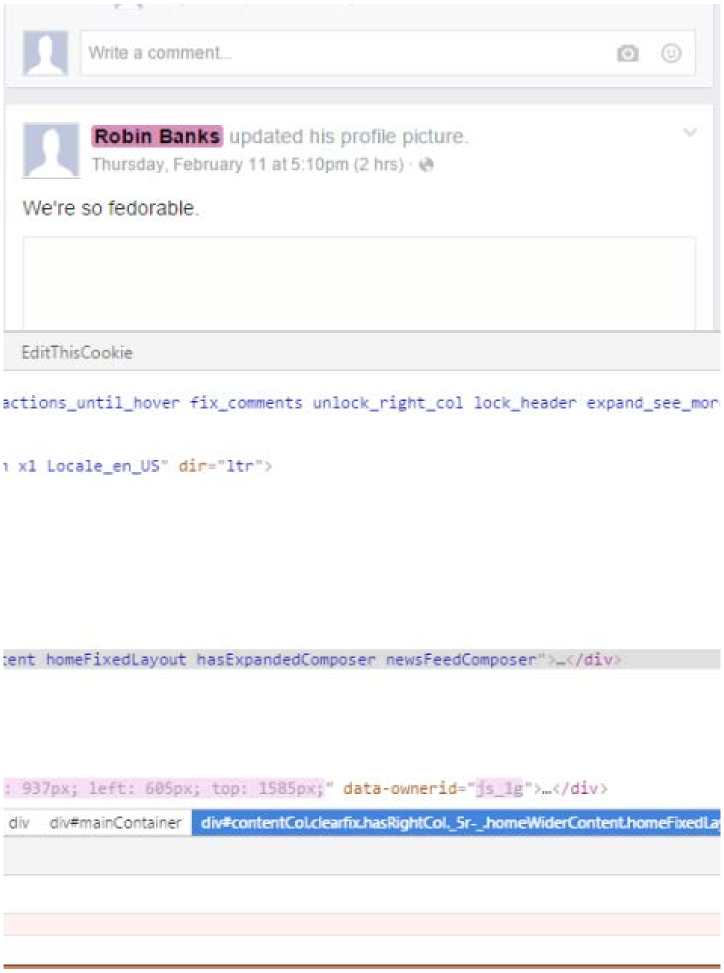


another user likes someone's link.

another user likes someone's link.

another user likes someone's post.

▲ **Figure 1:** Little green dots



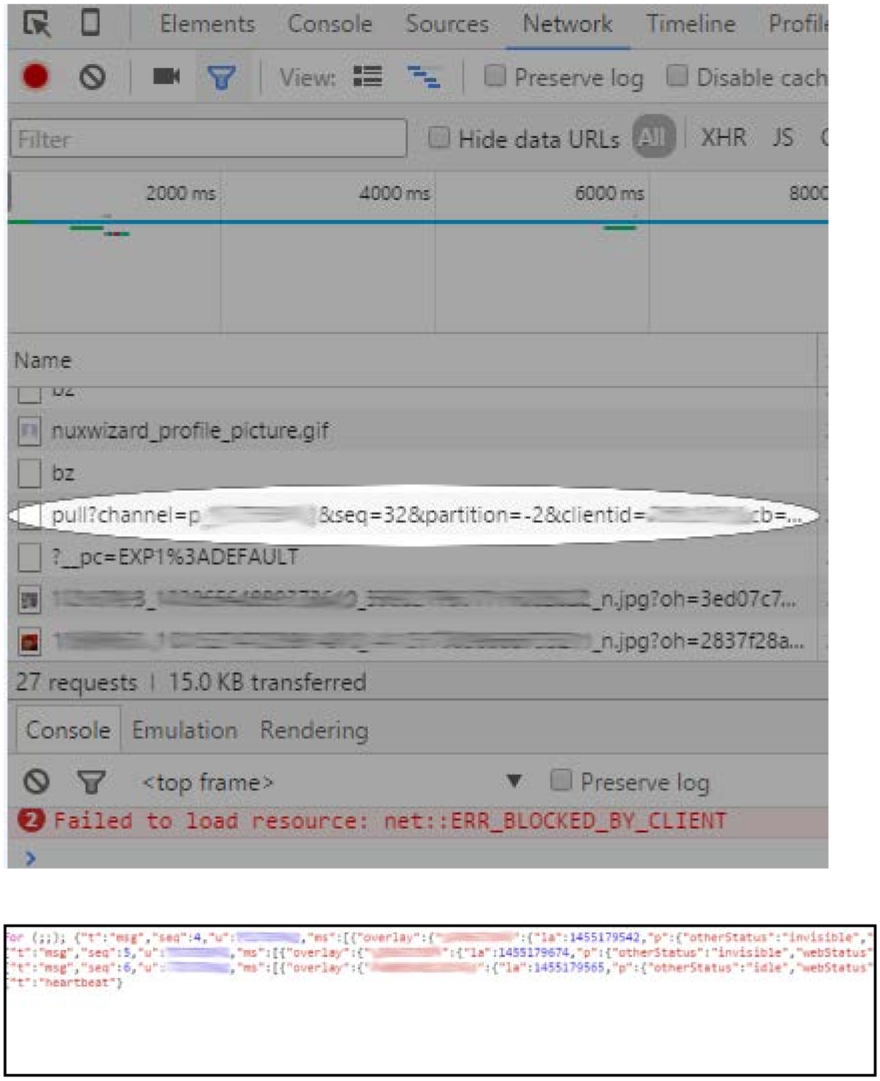
Robin Banks updated his profile picture.
Thursday, February 11 at 5:10pm (2 hrs) ·

We're so fedorable.

```
actions_until_hover fix_comments unlock_right_col lock_header expand_see_mor
ı x1 Locale_en_US" dir="ltr">
:ent homeFixedLayout hasExpandedComposer newsFeedComposer">…</div>
: 937px; left: 605px; top: 1585px;" data-ownerid="js_1g">…</div>
```

div
div#mainContainer
div#contentCol.clearfix.hasRightCol._5r-_homeWiderContent.homeFixedLa

▲ **Figure 2:** Inspect element



```
nuxwizard_profile_picture.gif
bz
pull?channel=p[redacted]&seq=32&partition=-2&clientid=[redacted]cb=...
?_pc=EXP1%3ADEFAULT
[redacted]_n.jpg?oh=3ed07c7...
[redacted]_n.jpg?oh=2837f28a...
27 requests | 15.0 KB transferred
```

```
2 Failed to load resource: net::ERR_BLOCKED_BY_CLIENT
```

```
for (;;); {"t":"msg","seq":4,"u":[redacted],"ms":[{"overlay":{"[redacted]":{"la":1455179542,"p":{"otherStatus":"invisible","
{"t":"msg","seq":5,"u":[redacted],"ms":[{"overlay":{"[redacted]":{"la":1455179674,"p":{"otherStatus":"invisible","webStatus"
{"t":"msg","seq":6,"u":[redacted],"ms":[{"overlay":{"[redacted]":{"la":1455179565,"p":{"otherStatus":"idle","webStatus"
{"t":"heartbeat"}
```

▲ **Figure 3:** Console snooping

- "webStatus": "idle" means their web browser is logged in to Facebook, and has the page open, but they aren't doing anything on the site like moving their mouse or talking to anyone.
- Since we have both of these at the same time, we can tell that this person is likely not using their phone, and that they were using facebook.com recently, but not right now.

That's already a little creepy that we can tell that about people. But can we do more with this?

You might also notice that there is a value called "la" that is a big integer that starts with "14." If you I dunno, didn't have a lot of friends in high school, you might recognise that as a UNIX time stamp — the time in seconds since midnight, January 1, 1970.

Computer scientists thought this would be a good time to start measuring the time from because the first app was born at midnight, January 1, 1970. The app was a custom emoji pack for an ancient model of phone that would one day evolve to become the first Blackberry.

If you're wondering why the response starts with "for (;;);", it's to, among other things, encourage developers to use a quality JSON decoder, instead of like, y'know, eval().

Anyway that "la" thing stands for "last active", and tells you the last time the person was active on Facebook, down to the second. Do you see where I'm going with this?

Roleplaying as the NSA

So far we have a whole bunch of things which look like this:

- A person
- A time
- Whether they're *online* or *offline* or *idle*
- Which devices they're online/offline/idle on

This doesn't seem that interesting at first, since you already know who is online by looking at the sidebar. But *what if there was someone always watching the little green dots?*

Using the *power of computers*, you can just write a Python program to listen to what the /pull requests are saying all the time ever, and write it down.

Here's a screenshot of all the log files I've got. (see Figure 4)

And here's what an individual log file looks like (the first 10 lines). (see Figure 5)

Those blurred out things are Facebook user ids. If you think these screenshots look *a little bit creepy* then YEAH I KNOW RIGHT.

Tell me about your program then, you massive nerd

It runs 24/7, and it’s constantly logging online/offline activity data from those /pull URLs using my Facebook cookie.

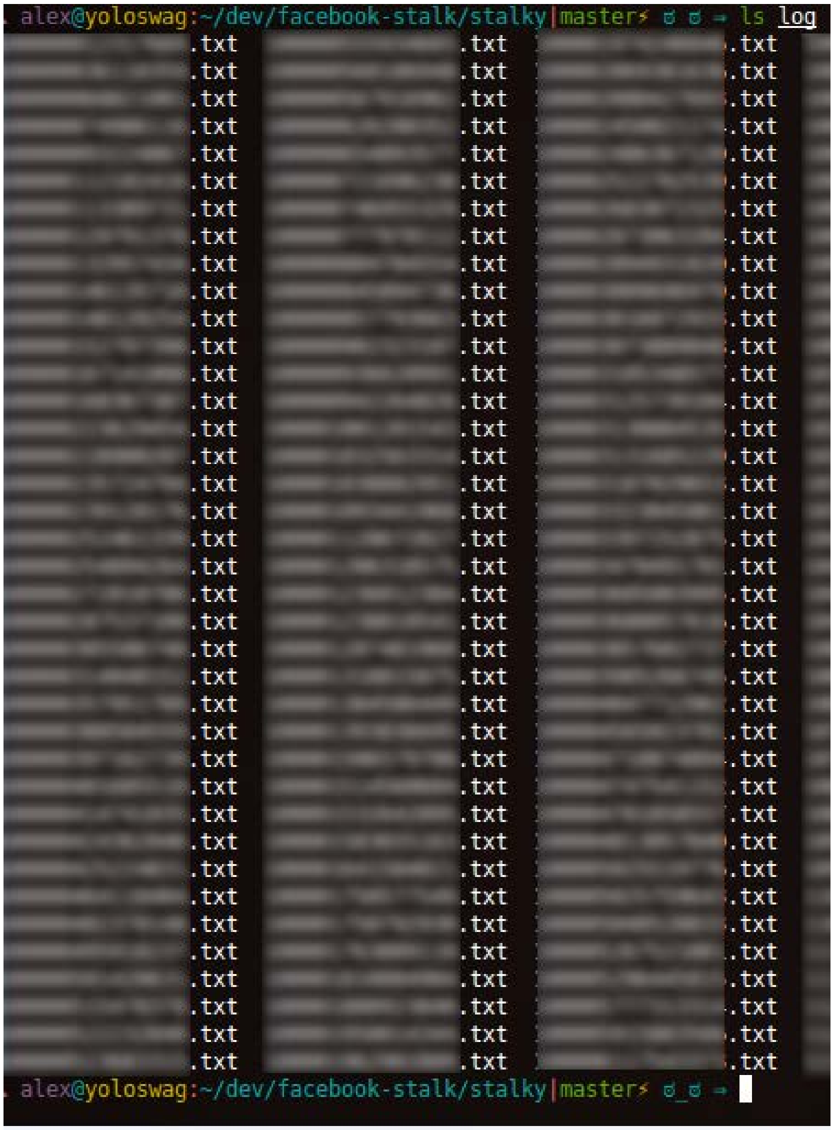
Writing it was *mostly* about saying “jeez, all these parameters look *complicated*” and then blindly copy/pasting them anyway.

Protip: you can right click on any network request in Chrome’s Developer Tools and click “Copy as cURL”. This is *amazing* and lets you re-run a request from the terminal, as well as give you all the headers and cookies used to run that request in a nice copy-pasteable format.

The first step was to just run that request verbatim in a terminal with curl.

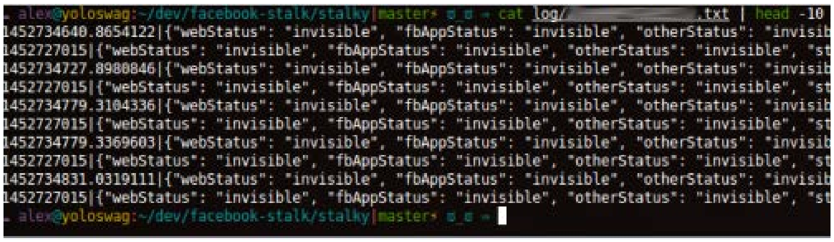
```
curl ‘https://1-edge-chat.
facebook.com/pull?channel=p_
[redacted]&seq=3&partition=-2&cli
entid=[redacted]&cb=6dcn&idle=5&
qp=y&cap=8&tur=1545&qpmade=14554
27171900&pws=fresh&isq=221841&ms
gs_recv=3&uid=[redacted]&viewer_
uid=[redacted]&sticky_
token=239&sticky_pool=atn2c06_
chat-proxy&state=active’ -H
'origin: https://www.facebook.
com’ -H 'dnt: 1’ -H 'accept-
encoding: gzip, deflate, sdch’
-H 'accept-language: en-
US,en;q=0.8,en-AU;q=0.6’ -H
'user-agent: ‘[redacted]’ -H
'accept: */*’ -H 'referer:
https://www.facebook.com/’ -H
'authority: 1-edge-chat.facebook.
com’ -H 'cookie: ‘[redacted]’ -
compressed
```

I was expecting it to not work because it looks like it has some sequence numbers in it, oh boy BUT it turned out to just



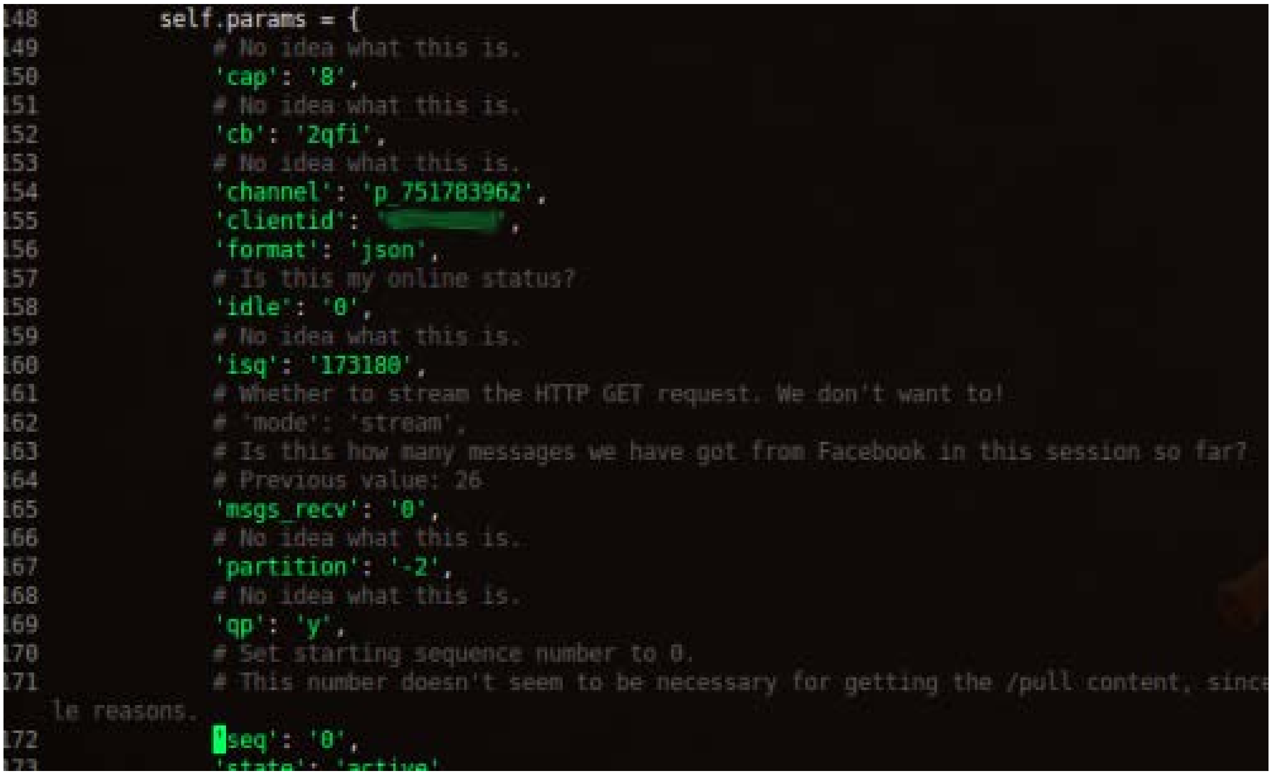
```
alex@yoloswag:~/dev/facebook-stalk/stalky|master⚡ ⊘ ⊘ → ls log
[redacted].txt    [redacted].txt    [redacted].txt
[redacted].txt    [redacted].txt    [redacted].txt
[redacted].txt    [redacted].txt    [redacted].txt
[redacted].txt    [redacted].txt    [redacted].txt
[redacted].txt    [redacted].txt    [redacted].txt
[redacted].txt    [redacted].txt    [redacted].txt
[redacted].txt    [redacted].txt    [redacted].txt
[redacted].txt    [redacted].txt    [redacted].txt
[redacted].txt    [redacted].txt    [redacted].txt
[redacted].txt    [redacted].txt    [redacted].txt
[redacted].txt    [redacted].txt    [redacted].txt
[redacted].txt    [redacted].txt    [redacted].txt
[redacted].txt    [redacted].txt    [redacted].txt
[redacted].txt    [redacted].txt    [redacted].txt
[redacted].txt    [redacted].txt    [redacted].txt
[redacted].txt    [redacted].txt    [redacted].txt
[redacted].txt    [redacted].txt    [redacted].txt
[redacted].txt    [redacted].txt    [redacted].txt
[redacted].txt    [redacted].txt    [redacted].txt
[redacted].txt    [redacted].txt    [redacted].txt
[redacted].txt    [redacted].txt    [redacted].txt
[redacted].txt    [redacted].txt    [redacted].txt
[redacted].txt    [redacted].txt    [redacted].txt
[redacted].txt    [redacted].txt    [redacted].txt
[redacted].txt    [redacted].txt    [redacted].txt
[redacted].txt    [redacted].txt    [redacted].txt
[redacted].txt    [redacted].txt    [redacted].txt
[redacted].txt    [redacted].txt    [redacted].txt
[redacted].txt    [redacted].txt    [redacted].txt
[redacted].txt    [redacted].txt    [redacted].txt
[redacted].txt    [redacted].txt    [redacted].txt
[redacted].txt    [redacted].txt    [redacted].txt
[redacted].txt    [redacted].txt    [redacted].txt
[redacted].txt    [redacted].txt    [redacted].txt
[redacted].txt    [redacted].txt    [redacted].txt
[redacted].txt    [redacted].txt    [redacted].txt
[redacted].txt    [redacted].txt    [redacted].txt
[redacted].txt    [redacted].txt    [redacted].txt
alex@yoloswag:~/dev/facebook-stalk/stalky|master⚡ ⊘ ⊘ →
```

▲ **Figure 4:** List of log files



```
alex@yoloswag:~/dev/facebook-stalk/stalky|master⚡ ⊘ ⊘ → cat log/[redacted].txt | head -10
1452734640.8654122|{"webStatus": "invisible", "fbAppStatus": "invisible", "otherStatus": "invisib
1452727015|{"webStatus": "invisible", "fbAppStatus": "invisible", "otherStatus": "invisible", "st
1452734727.8980846|{"webStatus": "invisible", "fbAppStatus": "invisible", "otherStatus": "invisib
1452727015|{"webStatus": "invisible", "fbAppStatus": "invisible", "otherStatus": "invisible", "st
1452734779.3104336|{"webStatus": "invisible", "fbAppStatus": "invisible", "otherStatus": "invisib
1452727015|{"webStatus": "invisible", "fbAppStatus": "invisible", "otherStatus": "invisible", "st
1452734779.3369603|{"webStatus": "invisible", "fbAppStatus": "invisible", "otherStatus": "invisib
1452727015|{"webStatus": "invisible", "fbAppStatus": "invisible", "otherStatus": "invisible", "st
1452734831.0319111|{"webStatus": "invisible", "fbAppStatus": "invisible", "otherStatus": "invisib
1452727015|{"webStatus": "invisible", "fbAppStatus": "invisible", "otherStatus": "invisible", "st
alex@yoloswag:~/dev/facebook-stalk/stalky|master⚡ ⊘ ⊘ →
```

▲ **Figure 5:** Individual log file (first 10 lines)



```
        self.params = {
            # No idea what this is.
            'cap': '8',
            # No idea what this is.
            'cb': '2qfi',
            # No idea what this is.
            'channel': 'p_751783962',
            'clientid': [redacted],
            'format': 'json',
            # Is this my online status?
            'idle': '0',
            # No idea what this is.
            'isq': '173180',
            # Whether to stream the HTTP GET request. We don't want to!
            # 'mode': 'stream',
            # Is this how many messages we have got from Facebook in this session so far?
            # Previous value: 26
            'msgs_recv': '0',
            # No idea what this is.
            'partition': '-2',
            # No idea what this is.
            'qp': 'y',
            # Set starting sequence number to 0.
            # This number doesn't seem to be necessary for getting the /pull content, since
le reasons.
            'seq': '0',
            'state': 'active'
```

▲ **Figure 6:** Data collection

take a really long time. I later found out this was because the /pull endpoint is using HTTP Long Polling, which turns out to be like a streaming HTTP GET request.

The only other important parameter to worry about is “seq,” which I’m guessing is the sequence number of the response from Facebook. Just add 1 to the sequence number that the response from /pull gives for the next request and you’re good to go.

If you’re worrying about remembering all this, chill out I got yo’ back — my 100% Terms of Service Compliant implementation of this is available here on GitHub. Standard disclaimers of “I’m so sorry I wrote parts of this in like 30 minutes” apply.

One caveat of the data-collection program that I’ve noticed is that it has false negatives. That is, sometimes it won’t give you a “this person is online” data point, even though they really are online. I guess that gives plausible deniability of… being offline?

You should probably get out more

[worried laughter]

So that’s the hard part done, right?

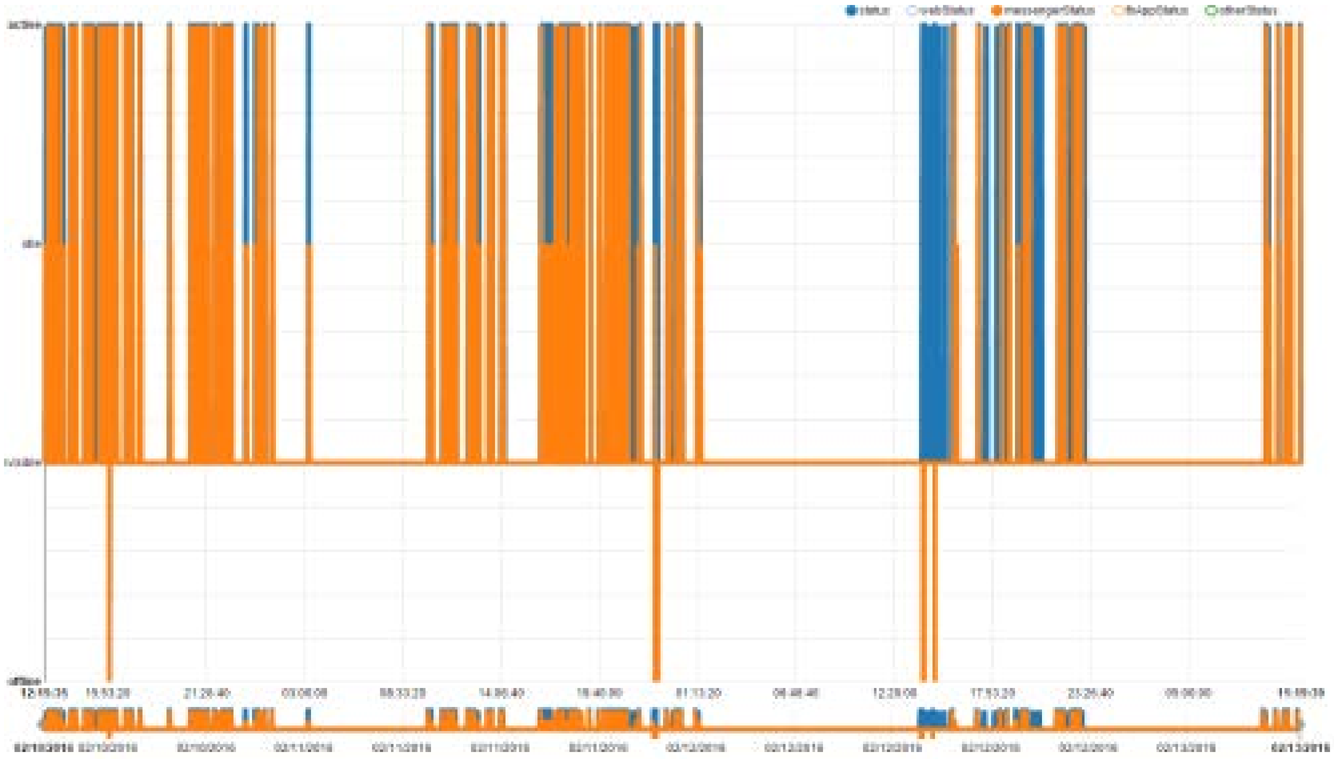
Let me paint you a word-picture. It’s 11pm, I’m listening to the soundtrack to The Social Network (ironically? meta-ironically? I don’t even know), I have six terminals tiled across two screens as well as fifty thousand browser tabs open and I’m up to my *third graphing library*.

Making graphs is really hard.

I used matplotlib, but I realised this wasn’t my thesis and I wouldn’t be embedding this ugly graph as a pdf into a LaTeX document that takes 3 passes of pdflatex to render because there’s been a terrible but extremely localised accident where only humanity’s LaTeX-to-pdf converters have been irreversibly sent back in time to the 80s.

I used bokeh, which claims to be a “matplotlib-killer”, and it was okay until a friend told me “it isn’t the 90s anymore, you don’t generate graphs server-side. Also your graphs are ugly and you should feel ugly *you utter fraud.*”

This friend recommended



▲ **Figure 7:** Person 1

nvd3.js, presumably because you're not making real graphs in 2016 unless your graphing library is <something>.js and requires at LEAST one other <something else>.js as a dependency. Everyone looks at you like "*what, you DON'T already use <something else>.js*? Jeez say goodbye to your Hacker News karma. Just apt-get install npm && npm install bower && bower install-" NO STOP IT THIS ISN'T WHAT TIM BERNERS-LEE WANTED."

I think it took about three times as much time to graph the data as it took to write the code to download it. *And the graphs aren't even good*! I gave up on perfecting the graphs so I could just hurry up and write this questionable blog post already. Just think of me resolving pip3 dependencies when you see the ugly graphs.

AND ANOTHER THING, when it's midnight and your x-axis formatting function doesn't convert UNIX times into JavaScript date objects properly because there's no timezone information and I dunno JavaScript was written by some guy in two weeks (yeah I ain't afraid to call it out what of it) and your binary-search based conversion of sparse timeseries data into uniformly dense timeseries data is causing *so many data points* to be graphed that it's slowly crashing Chrome and you're watching *helplessly* as your RAM goes up and Chrome won't close the tab and it just *doesn't seem right* that 2016, the year of the Linux Desktop has brought us this situation. I mean I thought if you had enough <something>.js libraries, this stuff was meant to just *scale right up* so tha.

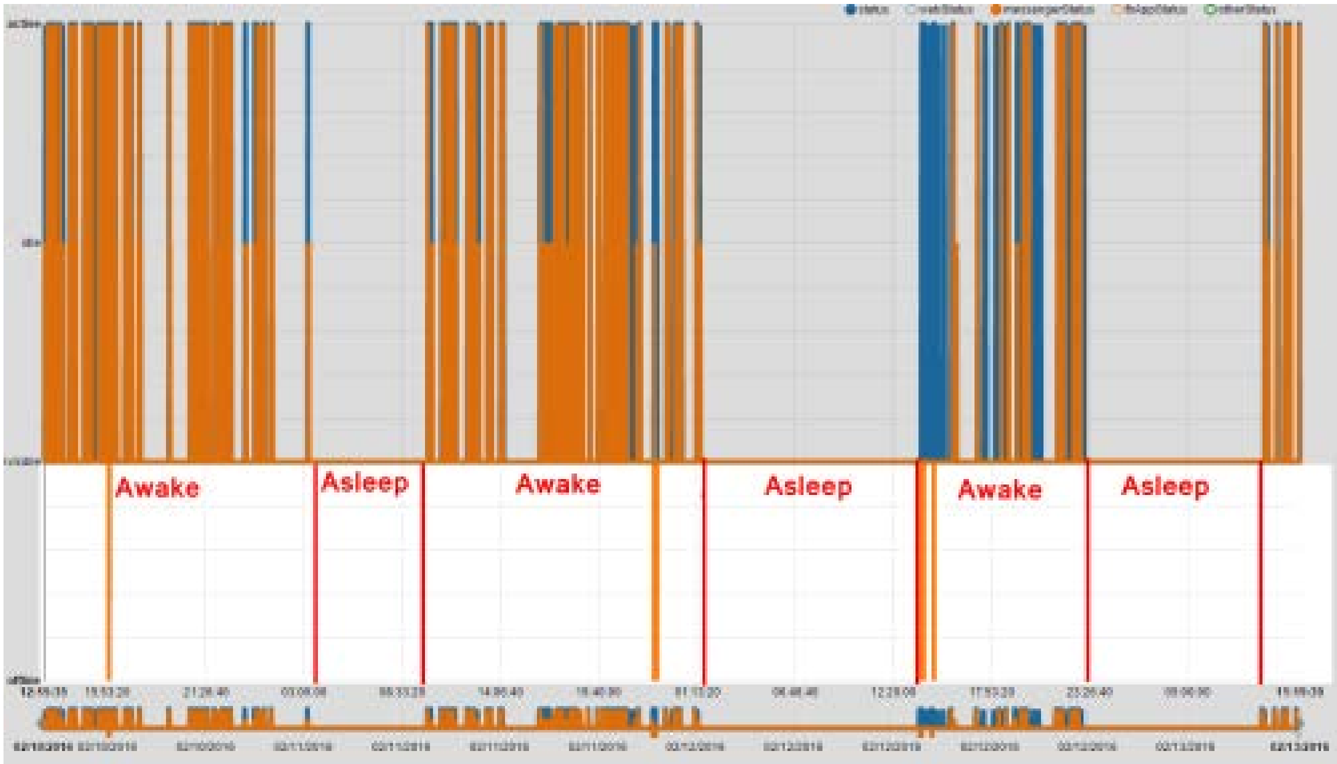
Quit stalling with graphing libraries and show me the graphs

Fine but you're missing out on top-quality graphing-related banter.

The graphs in this section are all of the online/offline activity of some of my Facebook friends.They consented to it being on this blog post on the condition that it's anonymous.

Person 1

Here's someone's graph. (see



▲ **Figure 8:** Person 1 with awake/asleep labels

Figure 7) The x-axis is time, and the y-axis is how online the user is. Possible states for someone's status are "offline," "invisible," "idle," and "active." Each coloured line is a different kind of client. It's called a client because I don't know I'm an Information Visualisation Professional and I get to make up words like that. Here are explanations for what each of the "coloured lines" means:

- *status* — Not sure what this is. Some kind of client-agnostic status? It doesn't line up exactly with the activity of the other clients though
- *webStatus* — Chat activity on facebook.com
- *messengerStatus* — Status on the Messenger mobile app
- *fbAppStatus* — Status on the Facebook mobile app
- *otherStatus* — Presumably shows when people are online on other apps that can access the API that causes them to be considered "online". OAuth? Random "apps" like Farmville? No idea.

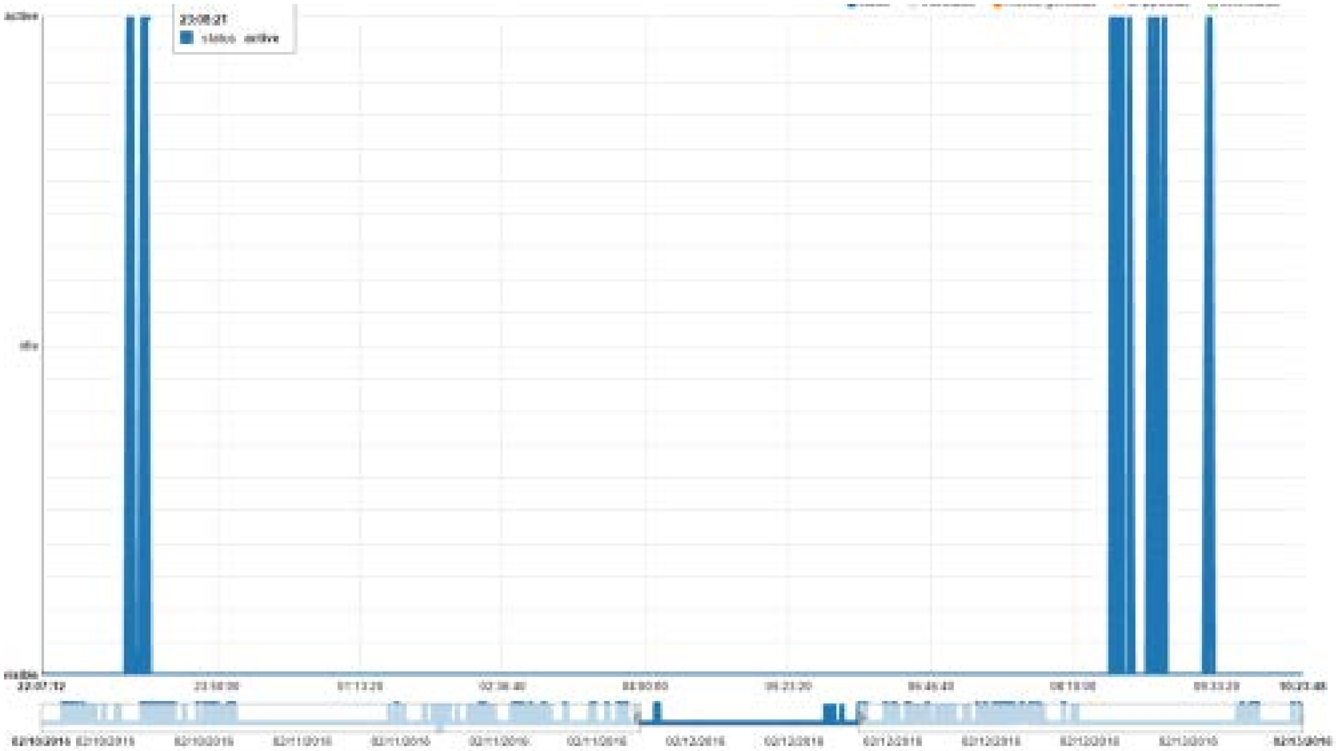
Here's the same graph, with some clumsy drawings on it showing when I think this person is awake/asleep. (see Figure 8)

You can see the amount of rest they're getting each day — it's the width of the "asleep" bit.

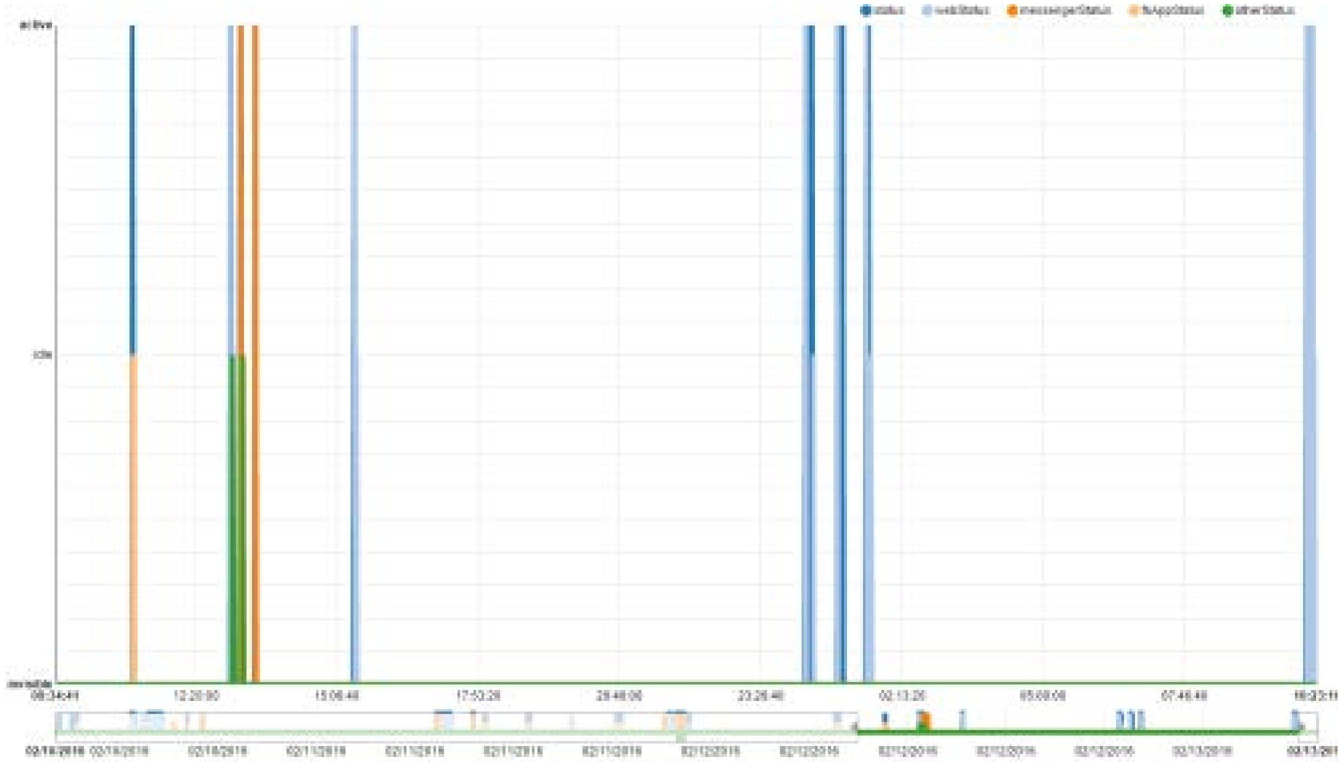
You can also see that they were probably asleep from 3am to 10am on February 11, and BOY does it feel creepy writing this.

Of course, this isn't perfect, since they might be awake and *not* using Facebook (I know). Having spoken to a few people who were graphed, it's been a fairly accurate measure of awake/asleep time, as well as "how much do you browse Facebook at work" time. ;)

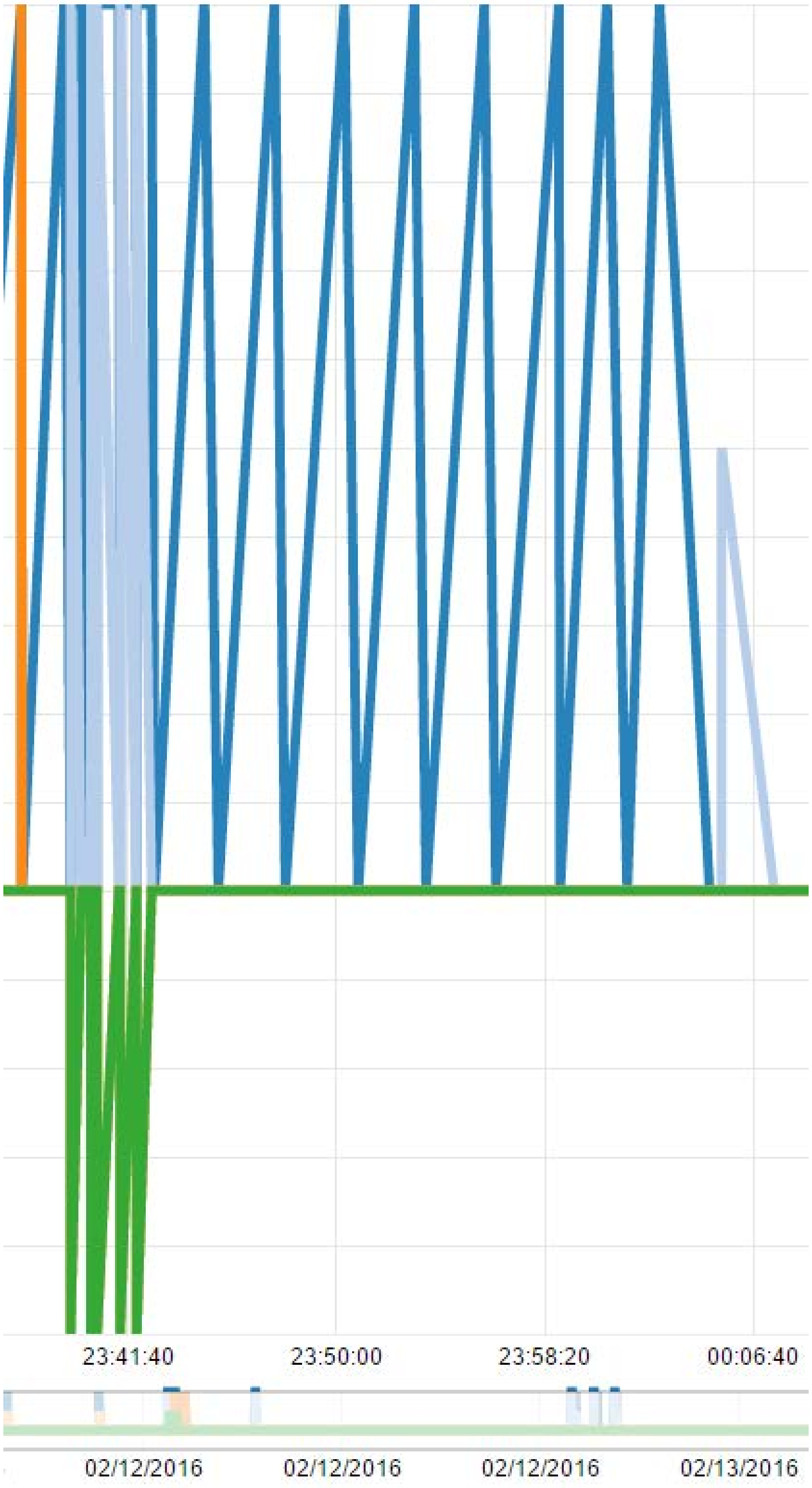
Do you look at Facebook shortly after you wake up? Shortly before you sleep? If so, these graphs are a fairly accurate way to measure when you were asleep, and anyone you're friends with on Facebook can do it.



▲ **Figure 9:** Person 2



▲ **Figure 10:** Person 3



▲ **Figure 11:** 3 minute spikes

Person 2

I showed this person their graph and asked them some questions. (see Figure 9)

> *"Did you go to sleep around 11:10pm last night?"*
>
> *They said yes.*
>
> *"Did you wake up around 8:32? That's a weird time. Was your alarm set for 8:30?"*
>
> *They said yes.*

NSA APPROVED.

Person 3

There are two interesting things about this graph. (see Figure 10)

- This person isn't online as frequently as the previous examples.
- This person isn't using the Messenger app nearly as much.

You can see that their webStatus was "online" on and off from midnight til around 2am, and then again at 10:21am.. I'm not sure if this spiky pattern means that they really were online, then offline, then online again, or if it's just a quirk of the dodgy undocumented "API" I'm using, or even if it's just a problem with my code.

Similarly, I'm not sure why there are these weird spikes every three minutes (+- ~1minute) sometimes. (see Figure 11)

Also, why does "otherStatus" go to offline precisely when "webStatus" goes to online? So many questions! Let me know if you know the answers to any of these things. (@Facebook employee friends)

Anyway, I hope I've convinced you that this is real creepy. I don't really want to be able to have the power to do this.

Your dumb graph screenshots are too small. Give me a live graph to play with

You got it, boss. Click here. Or anywhere, really. This whole sentence is a link.

What else can you do with this data?

You can aggregate. Finding the average wake up time/sleep time/time spent on Facebook each day and then looking for outliers sure sounds like a way to find interesting things about your Facebook friends.

You can write a thing to email you every morning with the names and sleep times of everyone who's had less than 6 hours of sleep.

You could even try and guess when your friends are talking to each other, by looking for times when only a few people are active, although I suspect this would be hard.

I'm sure you can come up with something else, too.

Why can you do this? Can't Facebook stop this from happening?

That's a good question, thanks for asking.

It makes sense for Facebook to be able to do this, since they can tell when everyone is online anyway. But why can your Facebook friends do this to you?

I don't know all the details of how facebook.com uses all the data that's sent via the /pull endpoint, but it's kinda creepy that I can see my friends' status on every device? I guess they could just give me "web" or "mobile" or "offline", rather than the full list of statuses for every client, but even that doesn't solve the problem.

I also see the value in seeing "last active 4h ago" and "last active 1m ago" for Messenger contacts but… I dunno, here I am making these creepy graphs.

Anyway, I just open-sourced my dodgy graph-making thing so now everyone can do this. And who knows how many people have been doing it already?

I'm probably oversimplifying it, though. The smart people at Facebook who write this stuff have probably thought through all of this and found that this way was best.

Can I stop you from doing this to me?

Kinda. Coincidentally, because my script is always running, collecting data, I show up as "online" all the time. If you were also running a script like this, it would partially prevent what I'm doing from working on you, since you always show up as "online", no matter what you're really doing. Activity from the Messenger app will still show up separately, though.

tl;dr

Facebook sends your computer a bunch of interesting information when you're on facebook.com.

You can collect that information over time and use it to keep track of when people are on Facebook, and which devices they're using.

You can make a pretty good guess as to what time people are going to sleep and waking up

It's creepy, but I don't see a way for Facebook to stop allowing this while still making their chat app good.

So how does this make money again?

Oh, no no no. I just, uh, don't get out much. ■

Reprinted with permission of the original author. First appeared at defaultnamehere.tumblr.com.

MAZUR
U.S. AIR FORCE

Building a startup in 45 minutes per day while deployed to Iraq

By MATT MAZUR

You may one day find yourself in a position where you're eager to work on a startup but limited by the amount of time you can put into it due to a day job, family or other obligations. In this post I would like to share with you all the story behind Lean Domain Search, a domain name generator that I built in about 45 minutes per day during a 5-month deployment to the Middle East. If you're struggling to find time to put into your startup, I hope this convinces you that you can accomplish a lot over time by putting a small amount of work into it each day.

Background

In the summer of 2011 I was a 26-year-old freshly pinned-on captain in the Air Force serving as a project manager at Hanscom Air Force Base in Massachusetts. I was 4 years into my 5-year service Academy commitment which meant that I had to serve one more year to pay back the Air Force for my education and training.

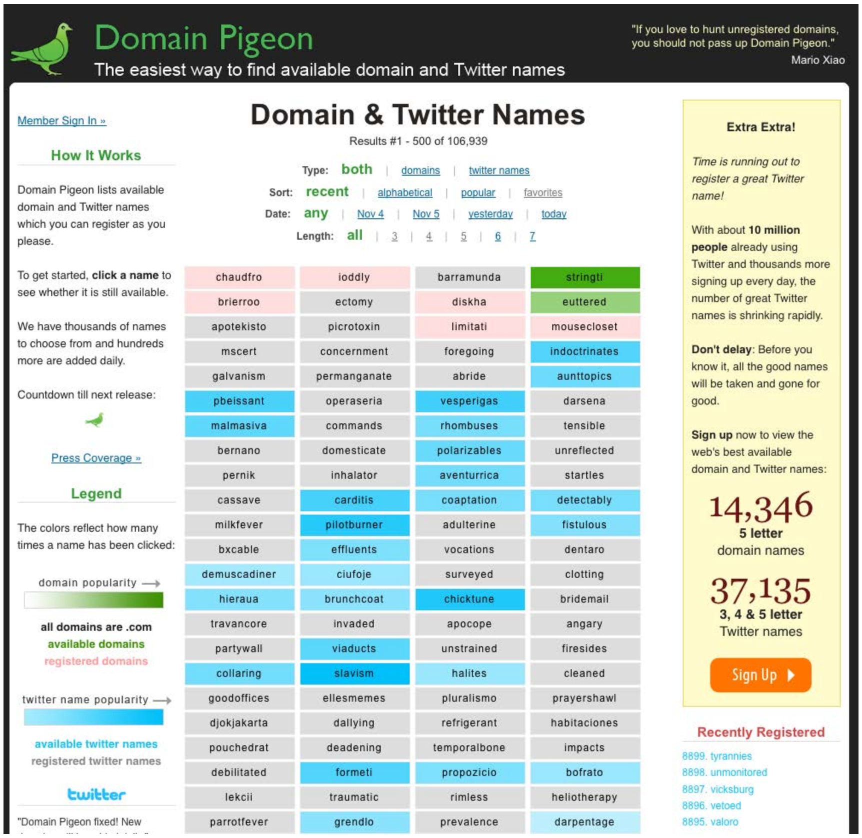
At the time I also had two moderately successful side projects that I had built on nights and weekends in the years prior: Preceden, a web-based timeline maker, and Lean Designs, a drag and drop web design tool.

Everything was going smoothly until my Unit Deployment Manager called me into his office one day and informed me that I had been selected to go on a six month deployment in August.

This presented quite a predicament. As a solo founder, I didn't have anyone I could turn my two projects over to maintain while I was away. I also had no idea what the Internet situation would be like wherever I was headed, but more importantly I didn't want to be distracted by these projects while I was out there.

I was contacted by the officer whose position I was going to take over when I arrived. He filled me in on some of the details and I eventually learned that there was limited internet access where I was going to live, but it was slow, had a firewall, and I'd probably be moving bases several weeks after I arrived anyway. I asked him if he could check to see if he had access to sites like Heroku (where my sites were hosted) and Github and he confirmed he did, but that still didn't guarantee I'd have access to make changes to my sites, time to work them, or even Internet access for the entire deployment.

I decided to keep the sites running, but to stop working on them several weeks prior to the deployment. That would provide time for any bugs to surface, which would allow me



Domain Pigeon

The easiest way to find available domain and Twitter names

"If you love to hunt unregistered domains, you should not pass up Domain Pigeon."
Mario Xiao

Member Sign In »

How It Works

Domain Pigeon lists available domain and Twitter names which you can register as you please.

To get started, **click a name** to see whether it is still available.

We have thousands of names to choose from and hundreds more are added daily.

Countdown till next release:

Press Coverage »

Legend

The colors reflect how many times a name has been clicked:

domain popularity →

all domains are .com
available domains
registered domains

twitter name popularity →

available twitter names
registered twitter names

twitter

"Domain Pigeon fixed! New

Domain & Twitter Names

Results #1 - 500 of 106,939

Type: **both** | domains | twitter names
Sort: **recent** | alphabetical | popular | favorites
Date: **any** | Nov 4 | Nov 5 | yesterday | today
Length: **all** | 3 | 4 | 5 | 6 | 7

chaudfro	ioddly	barramunda	stringti
brierroo	ectomy	diskha	euttered
apotekisto	picrotoxin	limitati	mousecloset
mscert	concernment	foregoing	indoctrinates
galvanism	permanganate	abride	aunttopics
pbeissant	operaseria	vesperigas	darsena
malmasiva	commands	rhombuses	tensible
bernano	domesticate	polarizables	unreflected
pernik	inhalator	aventurrica	startles
cassave	carditis	coaptation	detectably
milkfever	pilotburner	adulterine	fistulous
bxcable	effluents	vocations	dentaro
demuscadiner	ciufoje	surveyed	clotting
hieraua	brunchcoat	chicktune	bridemail
travancore	invaded	apocope	angary
partywall	viaducts	unstrained	firesides
collaring	slavism	halites	cleaned
goodoffices	ellesmemes	pluralismo	prayershawl
djokjakarta	dallying	refrigerant	habitaciones
pouchedrat	deadening	temporalbone	impacts
debilitated	formeti	propozicio	bofrato
lekcii	traumatic	rimless	heliotherapy
parrotfever	grendlo	prevalence	darpentage

Extra Extra!

Time is running out to register a great Twitter name!

With about **10 million people** already using Twitter and thousands more signing up every day, the number of great Twitter names is shrinking rapidly.

Don't delay: Before you know it, all the good names will be taken and gone for good.

Sign up now to view the web's best available domain and Twitter names:

14,346
5 letter
domain names

37,135
3, 4 & 5 letter
Twitter names

Recently Registered

8899. tyrannies
8898. unmonitored
8897. vicksburg
8896. vetoed
8895. valoro

▲ Lean Domain Search's predecessor — November 2010

to head out on the deployment knowing that the sites were in good shape. I also decided not to work on them at all during the deployment so that they wouldn't distract me from my job.

A small deployment side project

During the pre-deployment training one of our instructors suggested we pick up a hobby or something else to work on during downtime. For example, some officers use downtime during their deployments to take online classes towards a master's degree. I wasn't interested in that, but decided that I would try to work on a small software project when I had time.

Back in 2009 I had another domain search tool called Domain Pigeon. I was just getting started with web development so I couldn't figure out at the time how to do what I really wanted to do, which was to allow users to enter a keyword and pair the keyword with lots of other terms to generate and quickly check the availability of quality domains. Instead, I built Domain Pigeon, a service that simply listed interesting available .com domains.

I eventually shut Domain Pigeon down to focus on other projects, but the original idea stuck in the back of my head. By the time my deployment came around, I had a pretty good idea of how to implement it so I decided that would be what I would work on.

My daily schedule

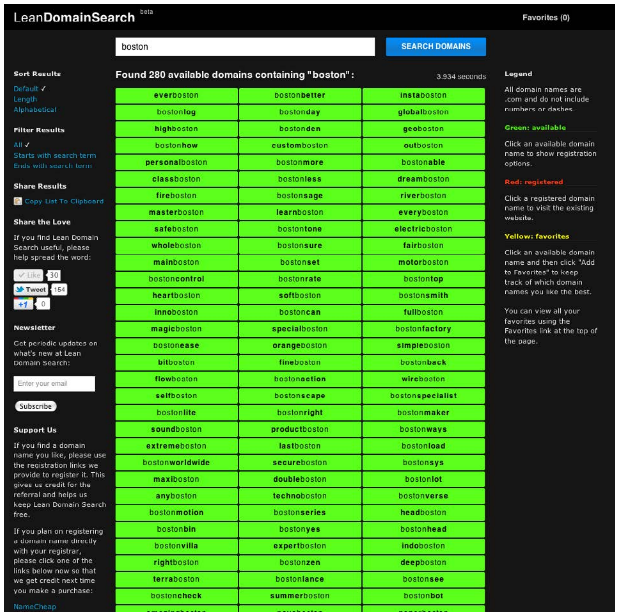
I wound up getting assigned to lead a team that oversaw communications (network, radio, satellite, etc) for the aviation unit that supported special operations forces in Iraq.

We worked 12-hour days every day for the entire deployment including weekends. I need roughly 8-9 hours of sleep to function at full capacity which left me with about 3-4 hours at the end of each day to have a meal, exercise, shower, chat with my wife, hang out with my coworkers, unwind and maybe work on my side project. In practice, that usually was about 45 minutes per day. Sometimes more, but often not at all.

Fortunately, there were never any major issues with my other projects during the deployment. A few small bugs surfaced, but nothing that impacted many users. I still had access to my email so I could respond to support requests when I had time. And because I was working on the new domain name generator locally on my laptop, I could work on it without worrying that there would be issues in production.

Piggy-backing on the popular lean startup movement as well as the name for my existing Lean Designs tool, I decided to call the new domain name generator Lean Domain Search.

Due to the drawdown of US forces in Iraq at the end of 2011, I wound up coming home after 5 months instead of six — in January 2012 instead of February 2012 like originally planned. I had two weeks of



Sort Results
Default ✓
Length
Alphabetical

Filter Results
All ✓
Starts with search term
Ends with search term

Share Results
Copy List To Clipboard

Share the Love
If you find Lean Domain Search useful, please help spread the word:

Newsletter
Get periodic updates on what's new at Lean Domain Search:

Support Us
If you find a domain name you like, please use the registration links we provide to register it. This gives us credit for the referral and helps us keep Lean Domain Search free.

If you plan on registering a domain name directly with your registrar, please click one of the links below now so that we get credit next time you make a purchase:

NameCheap

Found 280 available domains containing "boston": 3.934 seconds

everboston	bostonbetter	instaboston
bostonlog	bostonday	globalboston
highboston	bostonden	geoboston
bostonhow	customboston	outboston
personalboston	bostonmore	bostonable
classboston	bostonless	dreamboston
fireboston	bostonsage	riverboston
masterboston	learnboston	everyboston
safeboston	bostontone	electricboston
wholeboston	bostonsure	fairboston
mainboston	bostonset	motorboston
bostoncontrol	bostonrate	bostontop
heartboston	softboston	bostonsmith
innoboston	bostoncan	fullboston
magicboston	specialboston	bostonfactory
bostonease	orangeboston	simpleboston
bitboston	fineboston	bostonback
flowboston	bostonaction	wireboston
selfboston	bostonscape	bostonspecialist
bostonlite	bostonright	bostonmaker
soundboston	productboston	bostonways
extremeboston	lastboston	bostonload
bostonworldwide	secureboston	bostonsys
maxiboston	doubleboston	bostonlot
anyboston	technoboston	bostonverse
bostonmotion	bostonseries	headboston
bostonbin	bostonyes	bostonhead
bostonvilla	expertboston	indoboston
rightboston	bostonzen	deepboston
terraboston	bostonlance	bostonsee
bostoncheck	summerboston	bostonbot

Legend
All domain names are .com and do not include numbers or dashes.

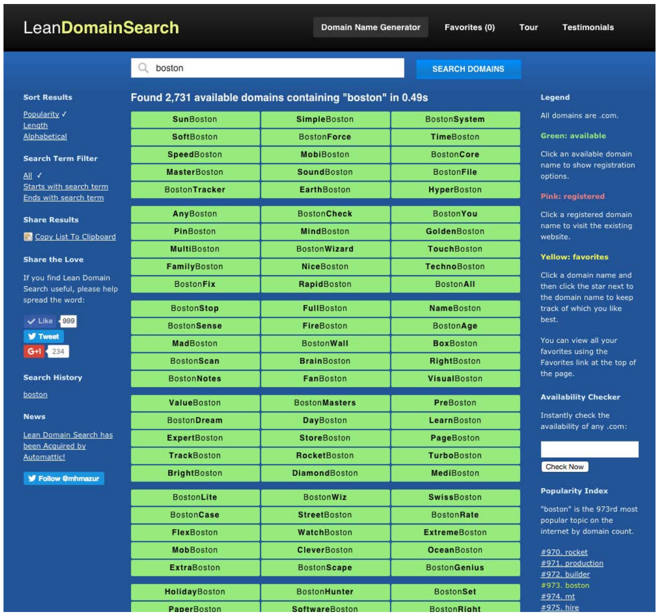
Green: available
Click an available domain name to show registration options.

Red: registered
Click a registered domain name to visit the existing website.

Yellow: favorites
Click an available domain name and then click "Add to Favorites" to keep track of which domain names you like the best.

You can view all your favorites using the Favorites link at the top of the page.

▲ Lean Domain Search — January 2012



Sort Results
Popularity ✓
Length
Alphabetical

Search Term Filter
All ✓
Starts with search term
Ends with search term

Share Results
Copy List To Clipboard

Share the Love
If you find Lean Domain Search useful, please help spread the word:

Search History
boston

News
Lean Domain Search has been Acquired by Automattic!

Found 2,731 available domains containing "boston" in 0.49s

SunBoston	SimpleBoston	BostonSystem
SoftBoston	BostonForce	TimeBoston
SpeedBoston	MobiBoston	BostonCore
MasterBoston	SoundBoston	BostonFile
BostonTracker	EarthBoston	HyperBoston
AnyBoston	BostonCheck	BostonYou
PinBoston	MindBoston	GoldenBoston
MultiBoston	BostonWizard	TouchBoston
FamilyBoston	NiceBoston	TechnoBoston
BostonFix	RapidBoston	BostonAll
BostonStop	FullBoston	NameBoston
BostonSense	FireBoston	BostonAge
MadBoston	BostonWall	BoxBoston
BostonScan	BrainBoston	RightBoston
BostonNotes	FanBoston	VisualBoston
ValueBoston	BostonMasters	PreBoston
BostonDream	DayBoston	LearnBoston
ExpertBoston	StoreBoston	PageBoston
TrackBoston	RocketBoston	TurboBoston
BrightBoston	DiamondBoston	MediBoston
BostonLite	BostonWiz	SwissBoston
BostonCase	StreetBoston	BostonRate
FlexBoston	WatchBoston	ExtremeBoston
MobBoston	CleverBoston	OceanBoston
ExtraBoston	BostonScape	BostonGenius
HolidayBoston	BostonHunter	BostonSet
PaperBoston	SoftwareBoston	BostonRight

Legend
All domains are .com.

Green: available
Click an available domain name to show registration options.

Pink: registered
Click a registered domain name to visit the existing website.

Yellow: favorites
Click a domain name and then click the star next to the domain name to keep track of which you like best.

You can view all your favorites using the Favorites link at the top of the page.

Availability Checker
Instantly check the availability of any .com:

Popularity Index
"boston" is the 973rd most popular topic on the internet by domain count.

#970. rocket
#971. production
#972. builder
#973. boston
#974. mt
#975. hire

▲ Lean Domain Search — Today

R&R after I got back, the first of which I spent with my wife on vacation in Maine, the second of which I launched the first version of Lean Domain Search.

I continued working as a project manager at Hanscom Air Force Base until my commitment ended in September 2012. My wife and I then moved back to Florida to be closer to family and I decided to work on Lean Domain Search full time.

Acquired

Remember I mentioned Domain Pigeon, my original domain name generator? When I launched it in early 2009, Matt Mullenweg, co-founder of WordPress and now CEO of Automattic, saw its launch on HackerNews and shot me an email saying he thought Domain Pigeon was neat and that if it didn't become a full time job there were a lot of opportunities to work on domains at Automattic.

We chatted briefly on Skype, but I was a second lieutenant at the time and still had over three years left on my Air Force commitment so it didn't go anywhere.

In early 2013 after I had been working on Lean Domain Search full time for several months, I remembered Matt's old email about Domain Pigeon. I checked out Automattic and WordPress.com and decided to reach back out to Matt to see if there was still an opportunity. I found his original email and responded to it again, this time 4 years after he sent it. I reminded him who I was, explained that I was working on a new domain name generator, and that I saw an opportunity for it to be put to use on WordPress.com to help users find better domain names.

He encouraged me to apply for a developer position which I did and in the end Automattic wound up hiring me and acquiring Lean Domain Search.

That period from August 2011 when I deployed to June 2013 when I started at Automattic was probably the most intense period of my life. I am extremely grateful that things worked out the way they did. In the end I wound up with a small acquisition, an amazing job at Automattic, a deployment that I'm really proud of, and experiences that will stay with me for the rest of my life.

If you're considering working on a startup but can't make the leap to do it full time for whatever reason, remember that even a few hours per week can have a huge impact in the long run.

Stick with it. Amazing things can happen. ■

Reprinted with permission of the original author. First appeared at mattmazur.com.

curl vs Wget

By DANIEL STENBERG

The main differences as I see them. Please consider my bias towards curl since after all, curl is my baby - but I contribute to Wget as well.

Please let me know if you have other thoughts or comments on this document.

File issues or pull-requests if you find problems or have improvements.

What both commands do

- both are command line tools that can download contents from FTP, HTTP and HTTPS
- both can send HTTP POST requests
- both support HTTP cookies
- both are designed to work without user interaction, like from within scripts
- both are fully open source and free software
- both projects were started in the 90s
- both support ***metalink***

How they differ

curl

- *library*. curl is powered by *libcurl* - a cross-platform library with a stable API that can be used by each and everyone. This difference is major since it creates a completely different attitude on how to do things internally. It is also slightly harder to make a library than a "mere" command line tool.
- *pipes*. curl works more lIke the traditional unix cat command, it sends more stuff to stdout, and reads more from stdin in a "everything is a pipe" manner. Wget is more like cp, using the same analogue.
- *Single shot.* curl is basically made to do single-shot transfers of data. It transfers just the URLs that the user specifies, and does not contain any recursive downloading logic nor any sort of HTML parser.
- *More protocols*. curl supports FTP, FTPS, Gopher, HTTP, HTTPS, SCP, SFTP, TFTP, TELNET, DICT, LDAP, LDAPS, FILE, POP3, IMAP, SMB/CIFS, SMTP, RTMP and RTSP. Wget only supports HTTP, HTTPS and FTP.
- *More portable*. curl builds and runs on lots of more platforms than wget. For example: OS/400, TPF and other more "exotic" platforms that aren't straight-forward unix clones.
- *More SSL libraries* and SSL support. curl can be built with one out of eleven (11!) different SSL/TLS libraries, and it offers more control and wider support for protocol details. curl supports public key pinning.
- *HTTP auth*. curl supports more HTTP authentication methods, especially over HTTP proxies: Basic, Digest, NTLM and Negotiate
- *SOCKS*. curl supports several SOCKS protocol versions for proxy access
- *Bidirectional*. curl offers upload and sending capabilities. Wget only offers plain HTTP POST support.

> ***Wget's major strong side compared to curl is its ability to download recursively...***

- HTTP multipart/form-data sending, which allows users to do HTTP "upload" and in general emulate browsers and do HTTP automation to a wider extent
- curl supports gzip and inflate Content-Encoding and does automatic *decompression*
- curl offers and performs decompression of *Transfer-Encoded HTTP*, wget doesn't
- curl supports *HTTP/2* and it does dual-stack connects using *Happy Eyeballs*
- *Much more developer activity*. While this can be debated, I consider three metrics here: mailing list activity, source code commit frequency and release frequency. Anyone following these two projects can see that the curl project has a lot higher pace in all these areas, and it has been so for 10+ years. Compare on openhub.

Wget

- Wget is *command line only*. There's no library.
- *Recursive!* Wget's major strong side compared to curl is its ability to download recursively, or even just download everything that is referred to from a remote resource, be it a HTML page or a FTP directory listing.
- *Older*. Wget has traces back to 1995, while curl can be tracked back no earlier than the end of 1996.
- *GPL*. Wget is 100% GPL v3. curl is MIT licensed.
- *GNU*. Wget is part of the GNU project and all copyrights are assigned to FSF. The curl project is entirely stand-alone and independent with no organization parenting at all with almost all copyrights owned by Daniel.
- Wget requires *no extra options* to simply download a remote URL to a local file, while curl requires -o or -O.
- Wget supports only *GnuTLS* or *OpenSSL* for SSL/TLS support
- Wget supports only *Basic* auth as the only auth type over HTTP proxy
- Wget has no SOCKS support
- Its ability to recover from a prematurely broken transfer and *continue downloading* has no counterpart in curl.
- Wget enables more features by default: cookies, redirect-following, time stamping from the remote resource etc. With curl most of those features need to be explicitly enabled.
- Wget can be typed in using only the left hand on a qwerty keyboard!

Additional stuff

Some have argued that I should compare uploading capabilities with wput, but that's a separate tool/project and I don't include that in this comparison.

Two other capable tools with similar feature set include aria2 and axel (dead project?) - try them out!

For a stricter feature by feature comparison (that also compares other similar tools), see the curl comparison table.

Thanks

Feedback and improvements by: Micah Cowan and Olemis Lang. ■

Reprinted with permission of the original author. First appeared at daniel.haxx.se.

Interesting

Work for only 3 hours a day, but every day

By NON UMEMOTO

I am an indie iPhone developer, and I've been working for 3 hours every day for almost 2 years now. It may not work for everybody, but I started this habit in early 2014, and I have continued to do it since have I found that this is the most productive way to work for me.

Taleb and DHH advices

I first got this idea when I watched the talk by DHH (Rails creator) in the startup school.

He was saying this:

> *"Working long hours isn't productive at all, if you work for 8 hours, try for 5 hours, or only for 4 hours. If you only have that time to work, you don't have time to see Twitter while working."*

Also, when I read the book Antifragile by Taleb, he mentioned that the trick to working in a productive way over a long period of time is to only work for a short amount of time every day.

Making money on the App Store is really tough, and people don't care how many hours I spend on my apps. They only care if it is useful or not. This is a completely result oriented world, but personally, I like it.

I have always thought about how I can optimize my time to work effectively, and after I tried a lot of different ways, I found it best to limit my work time each session for the best result in the long run.

Spaces are a very important factor in UI design, and that theory holds true for working.

Why 40 hours a week didn't work

I can choose how I spend my time since I am making my own apps, so first I'd been searching for the most effective way to divide my work time weekly and monthly.

There is no one who orders me to work, and I can rest anytime, so I made a quota first. For example, my first quota was 40 hours a week.

I calculated my work time using a stopwatch, and I noted things like "Ah, I worked for hours today", and "I went out yesterday, so I couldn't work, so let's work more today".

However, even if I work for the same amount of hours, the productivity depends on the conditions for each day. When I am tired or in a bad environment, I can't focus. The work quality was not consistent at all.

Often, even if I could focus for the first few hours, the more time would go on, the less I could focus.

Work short hours every day

Then, I made a rule to work only 3 hours every day without holidays. This is a bit extreme, but in this short hour limit, you are more motivated to work harder to make your working time meaningful.

First, the most productive time for me is after I wake up, so I need to sleep well, and start working right after I wake up. I don't read any news or SNS because even if I only read them a little bit, it could affect my productivity because it distracts my mind.

I even disable all notifications on my iPhone before I go to bed, so I don't see them before I start working next day.

I prepare for each day seriously like an athlete who prepares for their games in the morning. There was a huge difference of productivity between a 9-hour work day and a 3-hour day.

You really think about what to do

This was a good discovery. When you have only a short amount of time, you care about what you do more than ever.

When I develop features on my apps, I think more seriously if I should do it. Is it really worth my time for today? Is this project worth doing?

I cared about it before, but the seriousness increases when you have only a few hours to work a day.

Less stuck for coding

When you are coding, you get stuck quite often, and it can take a lot of hours to solve it sometimes. However, with my 3 hour work day, I find that this happens less since you can't keep digging into the issue when you don't have enough time anyway.

This way, you will be able to find the solution or come up with something the next day with a different viewpoint.

My challenge is that it is sometimes hard to go bed without solving some unknown issues, and you don't want to stop coding in the middle of it.

Nevertheless, when you take a break from the issues, you can think like "Well, it was not worth taking so much my time anyway…" in a calm mind the next day.

What if when you are in the zone?

Another pain for this method is that you should stop working anyway even when you are in the zone.

I often feel like I want to continue working when I am in the zone for some work. But, if you extend your work time rule once, you will do it again, then the more you extend, the more your productivity will drop.

It's a hard trade-off.

When you have only a short amount of time, you care about what you do more than ever.

> ***With this method, I don’t get stressed so much even if I keep working years...***

If I work for only a week, working more should produce more results, but when I work for a full month, the results from shorter work days will be more productive than if I was working longer days.

If I work for a year, I can complete my jobs more efficiently with this routine. I am sure I won’t retire after several years anyway.

Keep working until I die

Previously, I thought I would rather retire early and spend my life by having fun without working at all.

With this method, I don’t get stressed so much even if I keep working years, so I thought I could keep working and have fun until I die. This is the another surprising discovery I didn’t imagine before.

To stop working when I feel like I want to work more every day was the best way to keep working over a long period of time. It might allow me to keep running like a marathon runner with a same pace instead of working hard and retire early.

FAQ

I got a lot of emails after I posted this post, so I'll answer some popular questions here.

Q: I was wondering how work other than coding fits that profile. e.g. work with designer to prepare logo or any kind of promoting – that must be a part of your work as well, right?

A: Yes, I have to do everything, including UI & UX design, marketing, supporting and so on, since I’m a solo person. The coding might be around 50% of the work time.

Q: How do you monetize?

A: Free to use and In-App-Purchase for upgrading for Taxnote, Voicepaper, and Lisgo. ListTimer and Zeny are mostly ad-based. The revenue mostly comes from Japan now.

Q: Do you freelance, or are you available for hiring?

A: Not at the moment.

Q: What do you do with the remaining time?

A: I like reading and walking.

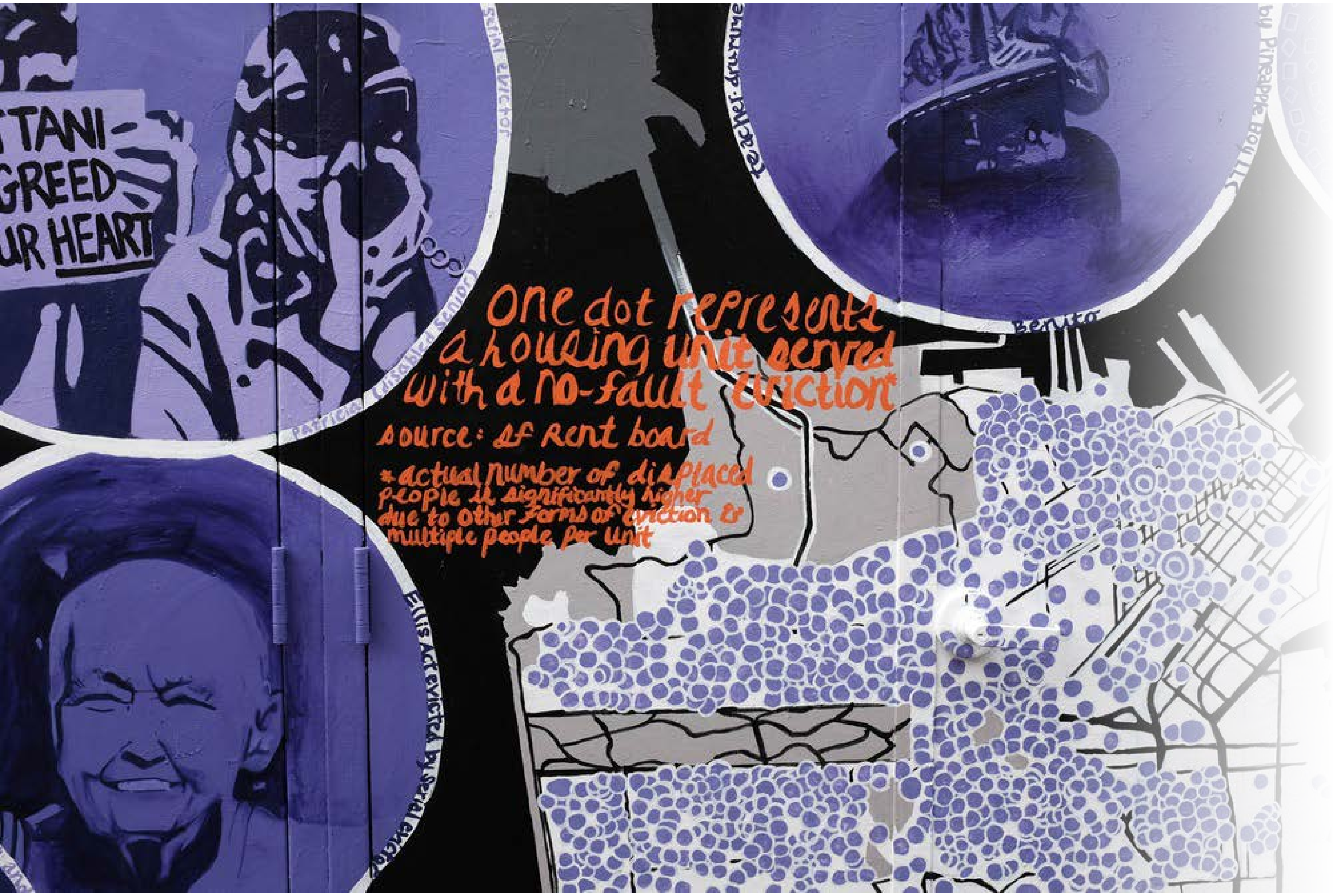
Q: Does it work for a freelancer?

A: Honestly, I don’t know, since I don’t have enough experience for that. I believe the best way to work depends on the situation and preferences for each person.

I might change my habit completely in the future, if I come up with a brilliant startup idea and want to work very hard on that every day.

I believe everyone has the right to choose how you use your time for the rest of your life. I consistently think about it too. ■

Reprinted with permission of the original author and plumshell.com.


TANI-
GREED
UR HEART
one dot represents a housing unit served with a no-fault eviction
source: SF Rent board
*actual number of displaced people is significantly higher due to other forms of eviction & multiple people per unit
Patricia (disabled senior)
Ellis Act evicted by serial evictor
teacher-drummer
Benito

MentalHealthError: An exception occurred

By KENNETH REITZ

The programming community has been opening up over the past few years about mental health issues. So, I want to take this opportunity to open up about my own.

Generally, my life has been extremely stable, with nothing peculiar of note. I've spent my time with friends/family, working on my hobbies (electronic music and synthesizers, photography), and primarily writing and maintaining a tremendous number of open source projects.

About a year and a half ago, however, things started to change. Once I began to recover from a debilitating consistent migraine, I had a newfound interest in yoga, meditation, and eastern philosophy. I had always been interested in different ways of thinking about the world, especially having been raised in a very religious household, so I spent a lot of time reading books by Ken Wilbur, Ram Dass, Terence McKenna, and Alex Grey. I started integrating many of these ideas into my primary worldview, which seemed perfectly normal and safe at the time.

> **I had a quartz crystal heart in my pocket which I was using to "channel the energy of the Internet."**

Fast-forward twelve months: I found myself in the Behavioral Health Department of Valley Health Medical Center for a "Voluntary Psychological Evaluation". While I was under the impression, going in, that I was free to check myself out at any time, the opposite was true — they weren't going to let me leave until I was well enough to leave.

I was not.

Sounds crazy, right? The so-called prolific Kenneth Reitz, of Requests fame, undergoing a required psychological evaluation? That only happens to other people.

Yeah, that's what I thought too.

I have bipolar affective disorder

This past September, I experienced what could be called a total mental health crisis event. During my hospitalization, I was diagnosed with "Bipolar Affective Disorder with Psychosis", which came as a complete shock to me — I'm almost always in an upbeat mood. It didn't make sense.

As it turns out, Bipolar Disorder isn't necessarily about having depression mood swings, as I once thought. It can take many forms, and gives many everyday people a great deal of struggle with operating in everyday life.

I have a few different phases my mind can go through, each like its own personality:

- **Normal:** standard-issue human.
- **Hypomanic:** extremely productive, increased confidence, very excited, very talkative, very awake (not tired).
- **Manic:** extreme version of hypomanic, total lack of inhibition, tremendous energy (sleep is impossible), often accompanied by hallucinations/psychosis.

Being hypomanic has always been fairly normal for me, and I credit most of my open source success to it. Sleep has never come easily to me while working on technical projects; I just don't get tired.

Being manic though, this one was new to me. My *crisis event* was caused by me going manic and not eating or sleeping for over four days (I was fasting). Looking back, I think this was my second manic episode — about a year earlier, I stayed up for a week for "spiritual reasons" while on a trip to Sweden (hallucinations and poor decisions followed).

This was my first serious manic episode.

When you're manic, you're the opposite of tired, and sleep is both undesired and impossible — it doesn't matter how long you've been awake. You want to avoid a manic state at all costs.

A painting of psychosis

I want to paint a picture of what the inside of my normal engineer's brain looked like during this crisis of psychosis. Be forewarned, I am normally a completely sane and normal human being, as I'm sure you know. What you're about to read is what can happen to anyone from a simple mental health issue. It's quite shocking.

Basically, I went *crazy*.

When I arrived at the hospital, I had experienced a number of hallucinations that caused me to believe that my world had a new set of rules that I needed to figure out. The experience was a lot like lucid dreaming, but in the real world. I was very confused.

I was under the impression that I was experiencing a "Kundalini Awakening," and did not require medical attention. I was aware that I was not acting normally, but I believed that I understood perfectly what was going on, while others did not.

I was having a severe identity crisis. When asked my weight, I struggled to answer "158 pounds" vs "the weight of the entire universe". When asked my name, I struggled between "Kenneth Reitz" vs. "I ॐ AM".

Due to an experience I had while hallucinating, I believed that every word I uttered became absolute truth, therefore I was extremely decisive with my words. People would ask me very simple questions and I would effectively have a very gentle panic attack.

I thought I had no emotions of my own, and all emotions I experienced were from people around me. My task was to breathe through these emotions, restoring the room to peace, and healing them.

I believed that I was experiencing multiple levels of reality at once — one where I was in the hospital, one where I was in prison, one where I was in heaven, one where I was in hell. I believed I was both completely alive and dead, asleep and awake, all at the same time, in a purgatory-like environment (the center of all dimensions).

I had no Internet access (or access to any technology), but I had a quartz crystal heart in my pocket that I was using to "channel the energy of the Internet."

Having studied philosophy, theology, and New Age woo-woo deeply over the past year, I cascaded through a number of theological self-identities. Each seemed like an inevitable truth that I was being constantly presented with, and forced to accept.

At first, I believed I was God (aka I ॐ AM THE BREATH OF LIFE). Then, Lucifer/the Serpent (Python!). Then, Narcissus. Then, Jesus. That seemed to upset other people. I then believed I was Archangel Metatron, and my task was to create the other angels. So, I spent an amount of time befriending other patients, and trying to show them how they were also angels. Then, I was Hermes/Mercury, the Messenger.

Once that trick wore off, I believed I was the Shaman of purgatory (the hospital), and I spent my time "holding the space" for the other patients, while they went into "ceremony" (group therapy). I was very conflicted at one point, because

> ***...I solved the puzzle, and realized that the simplest way to leave the hospital was to take the medicine the staff had been offering me the entire time...***

I believed I was created to do this, and I had the option to either stay in the hospital forever, raising Earth up into the stars, or quit and walk away (what the doctors wanted me to do). This was the most important decision in all of existence, and it had to be made immediately. Very stressful. After much thought, I thought of Genesis and how on the 7th day God rested. A good engineer doesn't need to keep his machine going manually forever; he just gets it going, then he goes home and rests.

I want to be a testament that this can happen to anyone you know, even you. It potentially already has. But, if so, you'll be fine in the end :)

About seven days in, my engineer brain started kicking back in. You may find this one disturbing, but I find it quite creative and entertaining. I believed that "KENNETH ROBERT REITZ"/METATRON was a trans-terrestrial being from the Sirius star system. My mind was the grand architect of all forms of physical and and consciousness technology, and I was responsible for improving the lives of everyone around me. My mind alone, for example, was responsible for the existience of the Pyramids of Egypt. Multiple times a day I would meditate my way between Earth and Earth's Sirius replica by basking in the Sun. The Earth represented an "ideal" logical volume of data (life), and the universe was the collective storage LUN. I was using Amazon's Dynamo algorithm to replicate life, with eventual consistency, throughout the universe by watering plants in the garden. Now, Amazon actually uses Requests to perform all internal API control operations for AWS, effectively making my code partially responsible for the operation of the Internet itself. See the theme? AWS US-EAST is located 70 miles away, which I considered to be America's version of the Pyramids of Egypt. The Earth was my Garden of Eden, and I wanted to go home to spend time with my Eve. I created this place for her, after all. I was very keen to have the doctors and my family look up the Dynamo white paper, to prove the legitimacy of my quest.

Keep in mind, I had been awake for about 10 days at this point, and still wasn't sure if I was alive or not.

As absolutely crazy as that sounds, that was my mind starting to recollect itself. I started to become aware of time and schedule patterns in the day. I was starting to identify less with theological absolutes and more with things closer to home: my own name, technology/code, and loved ones.

A breakthrough occurred when I slipped the doctor a piece of paper containing the URL to this website. This gave him a really good idea of who I actually was, which was a very useful tool in helping him diagnose me.

Once that wore off, I started to become more human again, focusing on being Kenneth and enjoying my time with the fellow patients. I went through many stages of identity conflict at this stage as well — I realized that I wanted to leave the hospital, and not stay there forever (as I originally wanted). I felt like I was in a puzzle, and one way of getting out was to become a doctor! So, I started walking up to the doctor (and all other levels of staff) and acted like a coworker, helping them do their job. At one point, I asked one of the nurses for a Direct Deposit Form (after seeing another patient with one), believing that was the key to establishing my employment. I was keen to inform them about my understanding of HIPPA compliance and the hospital's migration from an AS400 to a newer technology stack (EPIC).

Eventually, I solved the puzzle, and realized that the simplest way to leave the hospital was to take the medicine the staff had been offering me the entire time and get some sleep. At this point, I had been wide awake for 12+ days, and did not feel tired or sleepy once.

I finally left several days later, prescriptions and diagnosis in hand. It took me several weeks to completely come down from the trip, even with the heavy medication. I am tremendously thankful for the support of my family (and Heroku) during that time.

Thankfully, this was all back in September, and I'm 100% back to normal now.

How did this happen!?

A year prior, while getting into eastern religion and New Age philosophy, I started experiencing my first manic/psychosis symptoms after prolonged periods of meditation, which I was interpreting as spiritual events or "progress". Very real experiences, and they matched up with everything I was reading in books and online, so I thought I was really onto something.

I now believe that a great number of people within the ambiguously self-described "spiritual community" experience symptoms of mental illness. Kundalini yoga included. These communities, however, tend to view the symptoms as either positive effects, or far beyond the scope of standard medicine (doctors can't align a Bindu Chakra!).

Around the same time, right after having gone to my first Kundalini Yoga class, I ended up meeting (and falling in love with) a mesmerizing woman of mysticism that tenderly guided me off-the-deep-end with this style of thinking: numerology, synchronicity, Reiki, manifestation, the Mayan calendar, tarot, crystals, &c. My symptoms slowly got much worse. We shared a very deep and special bond. I heavily admired her, and felt I had much to learn from my newfound companion. We ended up spending nearly every day together, dating, taking trips all over the world, getting matching tattoos, performing thrill-seeking stunts, and attending shamanistic ceremonies together. We had an incredible time (the best year of my life), but there was a lot of unhealthy and certainly uncharacteristic behavior for me. Over the course of the manic year I spent with her, my delusional worldview (and hallucinations) had grown significantly worse, which led to the absurd themes of thinking featured in the above event.

The first time she left my apartment, I watched as a red/glowing infinitely-detailed sacred geometry adorned my plain white door. These are the types of hallucinations I would see on occasion, especially after prolonged periods of meditation or excitement. These experiences were interpreted to be of deep spiritual significance. Most hallucinations were non-visual, however, and involved subtle sensations best described in yogic terms as "feeling the flow of pranic energy". The rest could be described as an explosion of mental imagery with remarkable resolution/clarity.

How are you doing?

I am doing very well.

It's been about six months since this incident occurred, and I'm happy to say that I've made a full recovery. Bipolar Disorder is something I've had for a while (unknowingly), and will have for the rest of my life. I now know how to manage it, with the proper blend of awareness, medication, and sleep. It will always require extra special attention, though. It demands respect :)

Before, I was completely undiagnosed and had no idea there was even a problem. Going so long without a diagnosis also caused some very serious delu-

> ***It's been about six months since this incident occurred, and I'm happy to say that I've made a full recovery.***

sions to build, over time. That is unlikely to happen again, but I now know how to recognize any odd thought patterns and avoid psychological sinkholes if it does come up.

I also learned to rely on my family and friends to keep me in check and generally support my health as much as possible. I was a bit too self-sufficient before.

Before, I was completely undiagnosed and had no idea there was even a problem.

Now that I have a diagnosis, I have a much deeper understanding into the way my mind works, and know how to prevent another episode from occurring in the future.

In the past month, I've finally returned to actively contributing to my open source projects, for the first time since all of this started happening a year and a half ago.

I'm completely back to normal, before all the woo-woo entered my life, and I'm much happier and whole because of it. I'm completely grounded in material/physical/scientific reality, and it puzzles me that I could have ever not been this way. I still struggle with sleep occasionally, but I'm learning how to adapt to that.

As far as spirituality goes, I much prefer sticking to the absolute basics now — I eat, I breathe, I die. Spirituality 2.0 for Humans™!

I also got rid of my large collection of metaphysical books/tools. I still have a large collection of crystal spheres and skulls, but they look pretty cool on my desk :)

There's also a strange sense of relief that all those crazy things were due to a mental illness, and one which is pretty easy to control (now that I've been diagnosed).

I'm taking lithium now, and it seems to do a great job of keeping me in the normal/hypomanic range.

Conclusion

I wanted to share this story with you mostly because I thought you'd find it surprising. I haven't shared much, if anything, about this publicly, and I doubt others who have had similar experiences have either.

I want to be a testament that this can happen to anyone you know, even you. It potentially already has. But, if so, you'll be fine in the end :)

Personal takeaways

- Sleep is really important.
- This can happen to anyone, even you.
- Avoid falling in love with hyper-intelligent pan-dimensional beings. ■

Reprinted with permission of the original author. First appeared at kennethreitz.org.

food bit *

The uncomfortable truth about honey…

From sweetening foods to brewing mead and healing wounds, honey has been prized throughout history. But did you know that honey is essentially bee vomit? Sorry, but it's true. That stuff you've been stirring into your tea is formed when bees regurgitate the nectar they have collected onto the honeycomb. These industrious bees then flap their wings furiously over their, *ahem*, *undigested food*, to evaporate the excess water, resulting in the sweet sticky stuff that is universally loved.

* FOOD BIT is where we, enthusiasts of all edibles, sneak in a fun fact about food.

HACKER BITS is the monthly magazine that gives you the hottest technology and startup stories crowdsourced by the readers of Hacker News. We select from the top voted stories for you and publish them in an easy-to-read magazine format.

Get HACKER BITS delivered to your inbox every month! For more, visit hackerbits.com.